ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE LA SALUD

SENSORES BIOLÓGICOS SOBRE ARDUINO CON CONEXIÓN BLUETOOTH A APP ANDROID

ARDUINO BIOLOGICAL SENSOR WITH BLUETOOTH CONECTION TO ANDROID APP

Realizado por
Javier Mata Contreras
Tutorizado por
Rafael de Jesús Navas González
Departamento
Electrónica

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Febrero 2017

Fecha defensa:
El Secretario del Tribunal

Resumen: En el presente Trabajo Fin de Grado se expone una aplicación para terminales móviles que permite la adquisición y representación, entre otras funcionalidades, del electrocardiograma y el electromiograma del usuario. La adquisición de los datos biométricos está realizada mediante una arquitectura Arduino UNO Rev. 3 extendida con el sistema e-Health Sensor Shield V2.0. La comunicación entre el bloque de adquisición de datos y la aplicación móvil se realiza mediante el protocolo Bluetooth gracias a la extensión para Arduino Communication XBee Shield con el Bluetooth Module PRO. La aplicación móvil ha sido desarrollada sobre Android a través del entorno de desarrollo Android Studio. Dicha aplicación permite la representación gráfica de los datos obtenidos, así como su almacenamiento para su posterior visualización. Por otro lado, la aplicación también incorpora dos pequeños experimentos. El primero de ellos, consiste en el análisis de la modificación del ritmo cardiaco en función del tipo de música escuchada. El segundo, en el análisis de la intensidad media y duración de los impulso musculares.

Palabras claves:Android, Arduino, Bluetooth, e-Health, electromiograma, electrocardiograma

Abstract: This work presents a mobile device Android application with the goal of obtaining the electrocardiogram (ECG) and the electromyogram (EMG) bio signals and allowing their representation, between other functionalities. The biometric data acquisition is implemented by means of Arduino ONE Rev. 3 architecture, expanded with the e-Health Shield V2.0. The communication between data acquisition system and mobile application is achieved by Bluetooth Protocol, which is implemented using the Arduino Communication XBee Shield and the Bluetooth Module PRO. The mobile application has been developed using the Integrated Development Environment (IDE) Android Studio. The mobile application allows the ECG and EMG graphic representation, the data storage and its recovery for later visualization. On the other hand, two small experiments are also included. The first one, related with ECG, allows the relation analysis between heart rate and music that is heard by the user. The second one, related with EMG, allows the calculation of duration and mean intensity of the nervous impulse when the user muscle is contracted.

Keywords: Android, Arduino, Bluetooth, e-Health, electromyogram, electrocardiogram

Por tercera vez, a aquellos que se
alegran por esto más que yo.
En primero se fue mi padre
En segundo me casé
En tercero me mudé a una nueva ciudad
En cuarto el cuerpo me dio un aviso
Ha sido una aventura

AGRADECIMIENTOS

A lo largo de la aventura que ha sido para mí esta segunda Titulación Universitaria, hay muchas personas a las que debo un agradecimiento. En primer lugar a todos los docentes que me han aportado nuevos conocimientos y múltiples visiones sobre la docencia que me han hecho crecer como profesor. Y doblemente a aquellos que me han tenido como alumno a distancia y que han ayudado con amabilidad y comprensión a que no se notara el no poder asistir a sus clases. También a todos los compañeros que me han aceptado como uno más, aunque fuese el “abuelo” del aula, especialmente a Carmen y Fátima, que han sido mis ojos y mis oídos en la distancia.

A Rafael por aceptar ser mi tutor en el TFG y la ayuda ofrecida.

A Jose y Jorge, por realizar todas las tareas burocráticas y ser un apoyo constante.

A mi familia, que está lejos y cerca.

Y a Ana, por todo.

ÍNDICE

1

INTRODUCCIÓN

Este primer capítulo contiene la descripción del Trabajo Fin de Grado (TFG), tanto la motivación y objetivos del mismo como su organización y temporización. Igualmente, se detalla la estructura de la presente Memoria y sus diferentes apartados.

Finalmente, se incluyen diferentes subcapítulos donde se presentan aquellas ideas básicas para la comprensión de las herramientas utilizadas en el desarrollo del TFG. Se describen, de manera muy breve, las características básicas de los sistemas basados en arquitectura Arduino, así como sus extensiones e-Health y Comunication X-Bee+ Bluetooth PRO utilizadas para la captura de las señales biométricas y la comunicación inalámbrica. Por otro lado, se describe el entorno de desarrollo Android Studio, que permite la programación orientada a dispositivos móviles con Sistema Operativo Android.

Para finalizar, se describen, de manera simplificada, las características fundamentales de las bioseñales de interés para el TFG: El ElectroCardiograma y el ElectroMiograma.

1.1.- Motivación y Objetivos

En la actualidad, la automonitorización de señales biológicas se está extendiendo con rapidez entre la población, desde los clásicos pulsómetros hasta los medidores de glucosa que permiten a los diabéticos medir sus niveles de azúcar en su casa. Así mismo, la potencia y flexibilidad de los dispositivos móviles ha ido incorporando la gestión y procesado de estos datos de una manera sencilla y práctica. Paralelamente, la irrupción del sistema Arduino ha facilitado la creación y desarrollo de sistemas de adquisición de datos de manera relativamente barata y de fabricación personal.

Dentro de la Titulación de Grado de Ingeniería de la Salud, en su Mención de Ingeniería Biomédica se han tratado estas tres facetas. Tanto los conceptos básicos de la programación orientada a objetos (fundamental en la programación para dispositivos móviles), como en el diseño del *hardware* electrónico (Arduino), los conceptos relacionados con la adquisición y tratamiento de las señales biológicas y, lógicamente, el origen y características de éstas últimas.

Es con el objetivo de utilizar estos conocimientos y la motivación de profundizar en los conocimientos del diseño basado en Arduino y la programación orientada a dispositivos móviles donde se enmarca el presente Trabajo Fin de Grado (TFG). La propuesta consiste en la creación de una aplicación para dispositivos móviles que permita la adquisición y representación de las señales del Electrocardiograma (ECG) y el Electromiograma (EMG) de una persona. Estas señales serían adquiridas mediante un dispositivo basado en Arduino y su extensión (conocidas como “Shield”) e-Health y enviadas a través de una conexión Bluetooth, gracias a la extensión de Arduino Comunication XBee + Bluetooth PRO para comunicaciones inalámbricas vía Bluetooth.

La aplicación permitiría la representación gráfica de las señales adquiridas y su almacenamiento, permitiéndose su representación a posteriori para cualquier tipo de análisis. Por otro lado, la aplicación también incluiría, a modo de divertimento, dos pequeños experimentos que permitirían ahondar en el análisis de las señales del ECG y el EMG. De esta forma, en el primer caso, se presentará la funcionalidad de extraer el ritmo cardiaco a partir de ECG y analizar cómo éste se ve influenciado por el tipo de música que se escuche. En el segundo caso, se analizará el EMG para obtener la duración e intensidad del mismo, buscando que el usuario sea capaz de codificar distintas señales en función de la intensidad y duración de su movimiento.

1.2.- Organización de TFG y de la Memoria

Para cumplir el objetivo marcado en el epígrafe anterior, el trabajo realizado ha sido dividido en diferentes fases:

Fase 1: Adquisición de los conocimientos y materiales necesarios e instalación de las herramientas de desarrollo. Durante esta fase se han investigado los principios básicos de la programación del Arduino y de los dispositivos móviles con sistema operativo Android. Así mismo, se han buscado los suministradores de Arduino, así como de la extensión e-Health y las extensiones para las comunicaciones inalámbricas. Se han estudiado las distintas opciones y se han tomado las primeras decisiones respecto a cuáles eran las mejores para el TFG. Finalmente, también se han buscado e instalado el entorno de desarrollo para sistemas basados en Arduino y el Android Studio para el desarrollo de la aplicación móvil.

Fase 2: Desarrollo del sistema Arduino. En esta fase se ha desarrollado toda la arquitectura basada en Arduino la cual permite la captura de los datos de ECG y EMG gracias a la extensión e-Health. Los datos obtenidos son almacenados en ficheros de texto, lo que permite interactuar con la Fase 3, dado que estos ficheros con datos reales pueden representarse y procesarse en la aplicación móvil.

Fase 3: Desarrollo del sistema Android. En esta fase se ha desarrollado todo el interfaz de usuario de la aplicación móvil. Así mismo se han implementado todas las funcionalidades básicas, a excepción de la captura de las señales enviadas por los sensores. Para poder desarrollar la aplicación en su forma más cercana a la definitiva, se han usado ficheros de datos no solo para la representación, sino también como fuente de datos para emular la señal recibida de forma inalámbrica.

Fase 4: Desarrollo de la comunicación entre los sistemas Arduino y Android. Durante esta fase se ha desarrollado el interfaz de comunicación Bluetooth entre el sistema basado en Arduino y una aplicación móvil sobre Android, mediante unos programas sencillos.

Fase 5: Integración de los sistemas. En la fase de desarrollo final, se han incluido las comunicaciones entre los dos sistemas definitivos, permitiendo que la aplicación móvil obtenga las señales obtenidas por el sistema basado en Android de forma inalámbrica para su procesado y representación.

Fase 6: Escritura de la Memoria del TFG

Lógicamente, estas fases no se han realizado de manera disjunta, sino que se han ido solapando a lo largo del desarrollo del TFG. De igual manera, a lo largo de las Fases 2, 3, 4 y 5 se han ido incluyendo diferentes pruebas para verificar el correcto funcionamiento de las diferentes funcionalidades implementadas.

En el presente documento se muestra un resumen de todo este proceso de diseño, por lo cual se va a seguir una línea argumental muy similar a la organización del mismo. Se comienza con el presente capítulo introductorio en el que se describen la motivación y los objetivos del TFG, así como la organización del mismo. Un tercer bloque introduce, de manera muy breve, las características básicas y los conceptos relacionados con los sistemas que van a ser utilizados en el TFG (Arduino, Android, e-Health y Xbee). De igual forma se describen de manera muy somera las características de las dos señales biológicas analizadas (ECG y EMG) y las bases de su procesado.

El segundo capítulo se centra en el desarrollo del sensor de ECG y EMG basado en Arduino y e-Health. Se detalla la estructura general de conexión del *hardware* y la programación necesaria para la captura de los datos.

El tercer capítulo se dedica al desarrollo de la aplicación móvil en el entorno de desarrollo Android Studio. Se describe la estructura general de la aplicación y se detallan las distintas funcionalidades presentes, así como los detalles más importantes y relevantes al respecto de su programación con los fragmentos de código relacionados.

En el siguiente capítulo se desarrolla la estructura básica de la comunicación vía Bluetooth entre el sistema Arduino y la aplicación móvil mediante la creación de un sistema sencillo de envío y recepción de datos. Igualmente se describe las modificaciones necesarias a realizar tanto en la programación del Arduino como en la aplicación móvil Android para incorporar esta comunicación inalámbrica al sistema objeto del TFG, obteniéndose finalmente el sistema completo.

A lo largo de los diferentes capítulos se describe brevemente las diversas pruebas que se han ido realizando a lo largo del desarrollo del TFG con el objetivo de verificar el correcto funcionamiento de las diferentes partes del mismo, así como del sistema completo.

Finalmente, en un quinto capítulo se comentan las principales conclusiones a las que se ha llegado en el TFG y se adelantan algunas de las posibles líneas de mejora y estudio que pueden surgir a partir de la finalización del TFG.

1.3.- Conceptos Básicos

1.3.1. - Arduino

Arduino es un projecto de *hardware* y *software* de código abierto que se creó en el Instituto Ivrea de Italia en 2005 y, aunque existe una larga e interesantísima historia sobre su creación y derechos [1],[2], no es el objetivo de este documento narrarlas. Baste decir que se trata de un proyecto de fuente libre (open-source) consistente en un sistema *hardware* y un entorno de desarrollo (IDE) para su programación. Aunque actualmente se ofrece con el nombre de Genuino para fuera de USA, en el presente documento se va a continuar utilizando el nombre de Arduino, por el que sigue siendo muy conocido.

El sistema *hardware* consiste, en la mayoría de las versiones ofertadas, en un microcontrolador con varias entradas y salidas para señales analógicas y digitales y, generalmente, un puerto de comunicaciones USB para la conexión con un PC y la descarga de los programas a ejecutar.

Además de las diferentes placas básicas, existen multitud módulos y de extensiones, denominadas Shields, que permiten ampliar las funcionalidades básicas del Arduino, aumentando su potencialidad para su uso en cualquier tipo de aplicación (Fig.1.)

Fig.1. Diferentes productos Arduino (Fuente [3])

Para el presente TFG se ha optado por un Arduino UNO (revisión 3) como base. El Arduino UNO (Fig.2) es la versión de Arduino más utilizada y revisada y de la que se puede encontrar más información y ayuda para el diseño de aplicaciones. Está basado en el microcontrolador ATmega328P y consta de catorce entradas/salidas digitales (seis de ellas con posibilidad de utilizar modulación en anchura de pulso o PWM), seis entrada analógicas (no tiene salidas analógicas), un cristal de cuarzo de 16 MHz, un conector USB (que permite su alimentación y la conexión con un PC), un conector de alimentación (que permite su alimentación entre 7V y 12 V de manera autónoma mediante baterías externas o cargador AC/DC), un ICSP (In Circuit Serial Programming, que sirve para programar el BootLoader del Microcontrolador) y un botón de *reset* como características básicas. La posición de los componentes más importantes se puede ver en la Fig. 3.

Fig.2. Arduino(Genuino) UNO (Fuente [3])

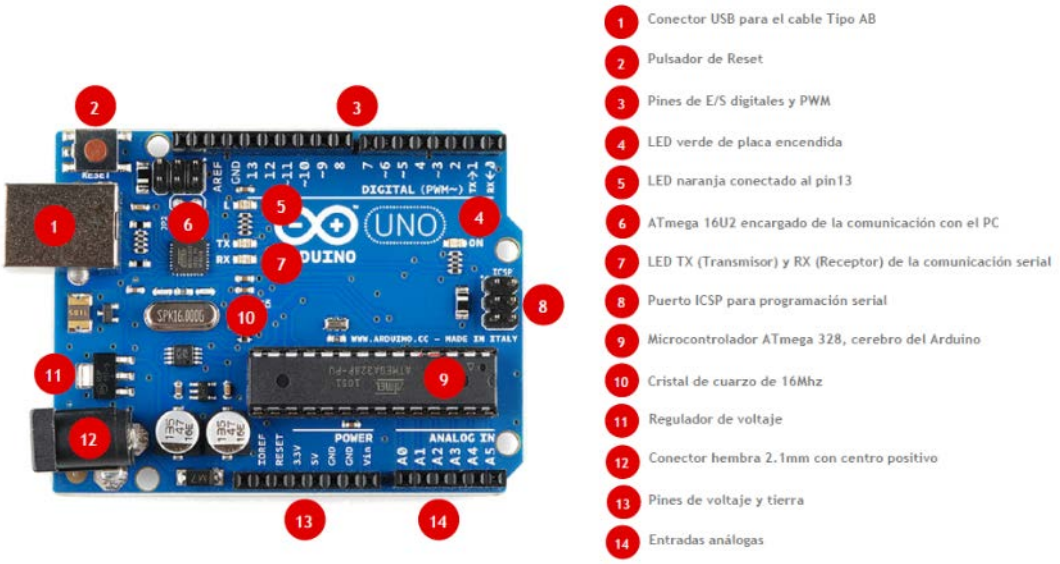


Fig.3. Componentes del Arduino(Genuino) UNO (Fuente [4])

Por otro lado, el lenguaje de programación utilizado para Arduino está basado en Wiring, que a su vez está basado en Java, mientras que el Entorno de Desarrollo (Arduino Software) está basado en Processing y puede ser descargado gratuitamente en https://www.arduino.cc/en/Main/Software. Su instalación es sencilla siguiendo las instrucciones mostradas en la propia página de descarga [4]. Los drivers para el Arduino UNO se instalan automáticamente al conectarlo por primera vez a través del cable USB al PC. Una vez instalado, se dispone de un editor de textos sencillo con las utilidades básicas de cualquier entorno de desarrollo *software*. Es necesario señalar que debe indicarse cuál es el tipo de Arduino para el que se desea compilar el programa una vez realizado, así como el puerto de comunicaciones que se va a usar. Estos detalles se especificarán con detenimiento en el capítulo 3, centrándose en lo necesario para el caso particular del presente TFG.

Respecto a los programas en Arduino propiamente dichos, éstos se denominan Sketchs y está compuesto de tres partes claramente diferenciadas (Fig. 4):

- Declaración de librerías y variables
- Setup()
- Loop()

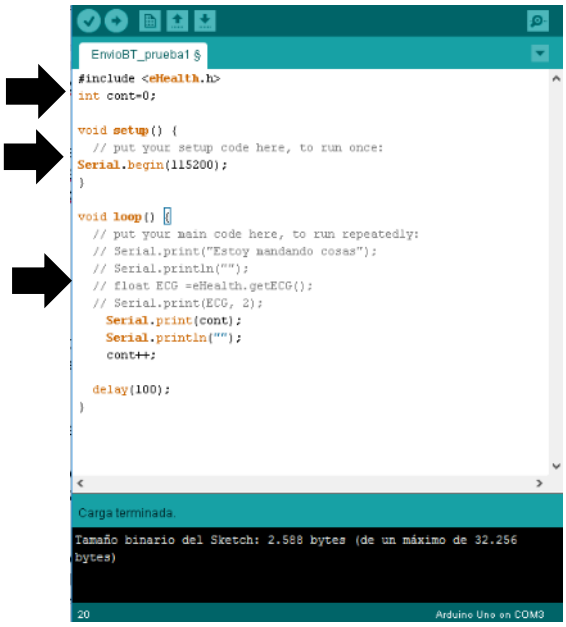


Fig.4. Sketch de Arduino.

En la declaración de librerías se puede incluir cualquiera de éstas, tanto ofrecidas por el propio IDE, como creadas por el propio programador o por terceros. Desde la versión 1.0.5 del Arduino Software, la inclusión de nuevas librerías es muy sencilla, pudiéndose hacer incluso de manera automática a

partir de un fichero .zip desde la opción de la barra de opciones *Sketch >> Include Library >>Add .ZIP* [4]. También incluirá la declaración de las variables que se consideren necesarias.

El bloque **setup()** incluye todas aquellas instrucciones que se ejecutarán una sola vez cuando el programa sea cargado o cuando se pulse el botón de *reset*. Suele contener la inicialización del sistema y de variables.

El bloque **loop()** forma un bucle que se repetirá indefinidamente una vez ejecutado el **setup()**. Contiene la programación propiamente dicha del Arduino.

De esta forma y mediante instrucciones sencillas es posible controlar todas las entradas y salidas del sistema e implementar cualquier funcionalidad deseada.

1.3.2. e-Health

Utilizando como base Arduino, han surgido numerosas extensiones que permiten ampliar las funcionalidades del mismo. Una de ellas es el Shield e-Health (Fig.5), pensado para la adquisición de señales biomédicas [5].

Fig.5. Extensión e-health con todos sus sensores (Fuente [5]).

En el presente TFG se ha utilizado el e-Health Sensor Platform V2.0, que en el momento de su inicio era la versión más actualizada del mismo. No obstante, a fecha de la escritura, Cooking Hacks, la marca de Libelum que comercializa los e-Health, ha lanzado una nueva versión denominada MySignals, con nuevos sensores y funcionalidades mejoradas y añadidas.

El e-Health Sensor Platform V2.0, además de para la medida del electrocardiograma (ECG) y el electromiograma (EMG), presenta conectores para otros seis sensores de señales biológicas. En total, las ocho señales biológicas tratadas son:

- Posición del Cuerpo
- Pulso y Oxímetro
- Presión Sanguínea
- Temperatura
- Flujo Respiratorio
- Respuesta Galvánica de la Piel
- Glucosímetro
- Electrocardiograma
- Electromiograma

La placa contiene toda la electrónica de acondicionamiento de la señal mientras que los sensores propiamente dichos han de ser adquiridos aparte. La alimentación de la extensión se realiza mediante una señal de DC de 12V / 2A que toma directamente de la placa del Arduino a la que se conecta. Dependiendo del uso que se le dé y del PC al que esté conectado, si la alimentación del Arduino es a través del USB puede no ser suficiente, necesitándose una fuente de alimentación externa [5]. En el caso de los sensores de ECG y EMG y el PC utilizado no ha sido necesaria ninguna alimentación adicional.

Por otro lado, como se puede observar en la Fig. 6, la extensión permite acceder de manera transparente a la mayoría de los conectores de entrada y de salida del Arduino UNO, lo que habilita la utilización de estos para cualquier funcionalidad añadida (como se verá, esto es importante al querer incorporar también la extensión de comunicaciones). Además, ofrece la posibilidad de conexión de una pantalla LCD para la representación de los resultados (no utilizada en este TFG).

Los sensores de ECG y EMG son físicamente muy parecidos (Fig. 7a y Fig. 7b, respectivamente), diferenciándose fundamentalmente en el tamaño de los parches del electrodo. No obstante, son conectados a distintos terminales y ha de seleccionarse mediante un puente cuál de los dos es utilizado para medir.

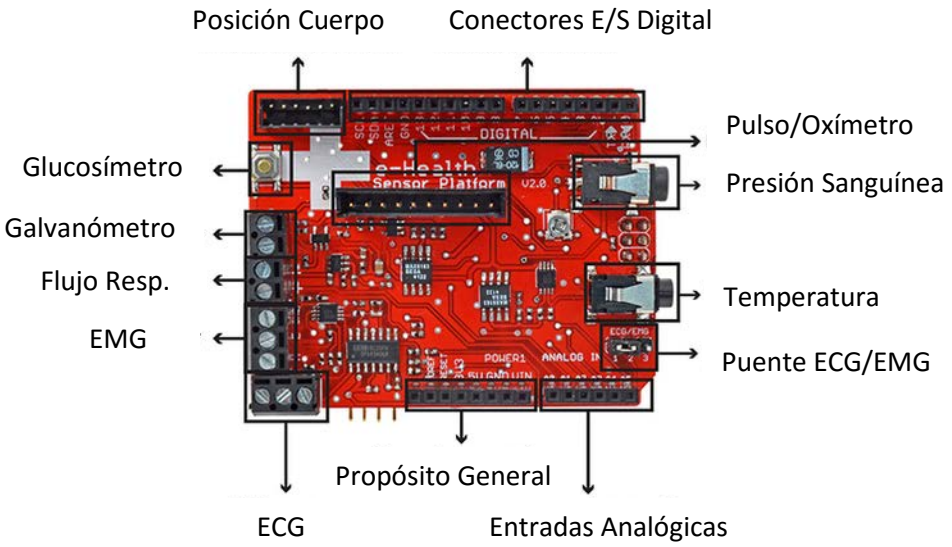


Fig.6. Conectores principales de la extensión e-Health.

(a) (b)

Fig.7. Cables y electrodos para ECG (a) y para EMG (b)

El resto de detalles específicos de esta extensión necesarios para el presente TFG serán desarrollados en el capítulo 2.

1.3.3. Comunication-X-Bee Shield + Bluetooth Module PRO

Otra de las extensiones que es posible encontrar para Arduino es el Communication Shield X-Bee para comunicaciones inalámbricas (WiFi, RFID, Bluetooth o XBee). Desarrollada, al igual que el e-Health, por Libelum, también ha evolucionado en una nueva versión denominada Wireless Shield, aunque en el presente TFG se utilizará la versión antigua, única disponible al inicio del desarrollo de mismo. A esta extensión se le pueden incorporar diferentes módulos en función del protocolo de comunicaciones que se desee utilizar. En el caso tratado aquí, y como ya ha sido comentado, se ha optado por el

protocolo Bluetooth, debido a su simplicidad y a la amplia documentación encontrada, utilizando para ello el módulo Bluetooth PRO, del mismo fabricante que la extensión y con conexionado compatible con el XBee (Fig.8), para poder conectarlo a la placa de comunicaciones XBee.

Fig.8. Módulo Bluetooth PRO montado sobre Shield de comunicaciones XBee a su vez montado sobre Arduino ONE.

La extensión Communication XBee cubre los pines analógicos y parte de los digitales, aunque permite el acceso de manera transparente a la mayoría de estos, de igual forma que el e-Health hacía con los del Arduino ONE, con lo pueden ser montados unos sobre otros.

Será muy importante, como se verá en el capítulo cuatro, la existencia de un puente que permite configurarlo de dos formas: posición XBEE y posición USB. En la primera de ellas los datos enviados lo son tanto por el módulo inalámbrico que se haya conectado, como por el conector USB (a efectos de la placa de Arduino, ambos son considerados como el puerto serie. Sin embargo, como puerto de entrada solo se considera el dispositivo inalámbrico, no pudiéndosele mandar comandos al Arduino a través del puerto USB (los comandos llegarán directamente al módulo Bluetooth). En la segunda posición, el módulo de comunicación inalámbrico puede comunicarse directamente con un PC a través de puerto USB, siempre que el microcontrolador de la placa Arduino haya sido retirado. En otro caso, la comunicación siempre será entre PC y microcontrolador (como si el bloque de comunicaciones no estuviera) [6].

Para concluir esta breve exposición de los diferentes componentes utilizados, indicar que el módulo Bluetooth PRO incorpora un Chip Bluegiga WT12, que implementa el protocolo de comunicaciones Bluetooth v2.1 + EDR.

Class2, además del firmware denominado Iwrap. Tiene siete niveles de potencia transmitida [-27dBm, +3dBm] y una sensibilidad en recepción de -90 dBm. Incorpora, así mismo, una antena de 2dBi (prácticamente omnidireccional) [7].

1.3.4. Android Studio

Para el desarrollo de la aplicación en el terminal móvil se ha optado por aquellas que utilizan el sistema Android. Para ello, se ha hecho uso del entorno de desarrollo Android Studio, basado en IntelliJ IDEA (un IDE para Java) [8]. Sus principales características son su sistema de compilación flexible basado en Gradle; emulador rápido con varias funciones junto a la posibilidad de usar un dispositivo con Android para ejecutar y depurar directamente sobre él, vía conector USB y también la facilidad de poder realizar desarrollos para todos los dispositivos Android en un entorno unificado.

Los programas desarrollados con Android Studio están compuestos por uno o más módulos, cada uno de los cuales contiene tres bloques principales: Manifiestos, que consiste en el archivo AndroidManifest.xml; Carpeta Java, que contiene todos los archivos de fuente en java; y Carpeta Res, que contiene los recursos (diseños XML, imágenes, estilos, etc).

El AndroidManifest.xml es un fichero que contiene una descripción de las características de la aplicación (versión del DSK de Java, actividades que contiene, librerías, configuraciones, permisos, etc).

En la carpeta Java se encontrará el código fuente de todas las actividades que componen la aplicación. Es donde se encuentra realmente programada toda la funcionalidad que se desea realizar. Cada actividad es un conjunto de acciones e interacciones que se pueden ejecutar por el usuario en primer plano y que siguen un ciclo de vida, desde su creación hasta su destrucción (Fig. 9). Una actividad puede pausarse o pararse en mitad de su ejecución (la diferencia radica en cómo se reanudaría la actividad, con un reinicio completo o no), pudiéndose, en el caso de parar, lanzar una nueva actividad, que pasará ahora a primer plano [9].

Finalmente, en la carpeta Res se encuentran todos los recursos de la aplicación (imágenes, sonidos, estilos, textos…). Una de las subcarpetas más importantes de esta sección es la carpeta Layout. En ella se encuentra descrito, en código XML, el interfaz de usuario de cada una de las actividades que compongan la aplicación. En el caso de Android Studio, estos layouts pueden ser diseñados de manera sencilla mediante un interfaz gráfico que permite seleccionar y posicionar los componentes gráficos que se consideren necesarios para la aplicación, así como modificar sus atributos.

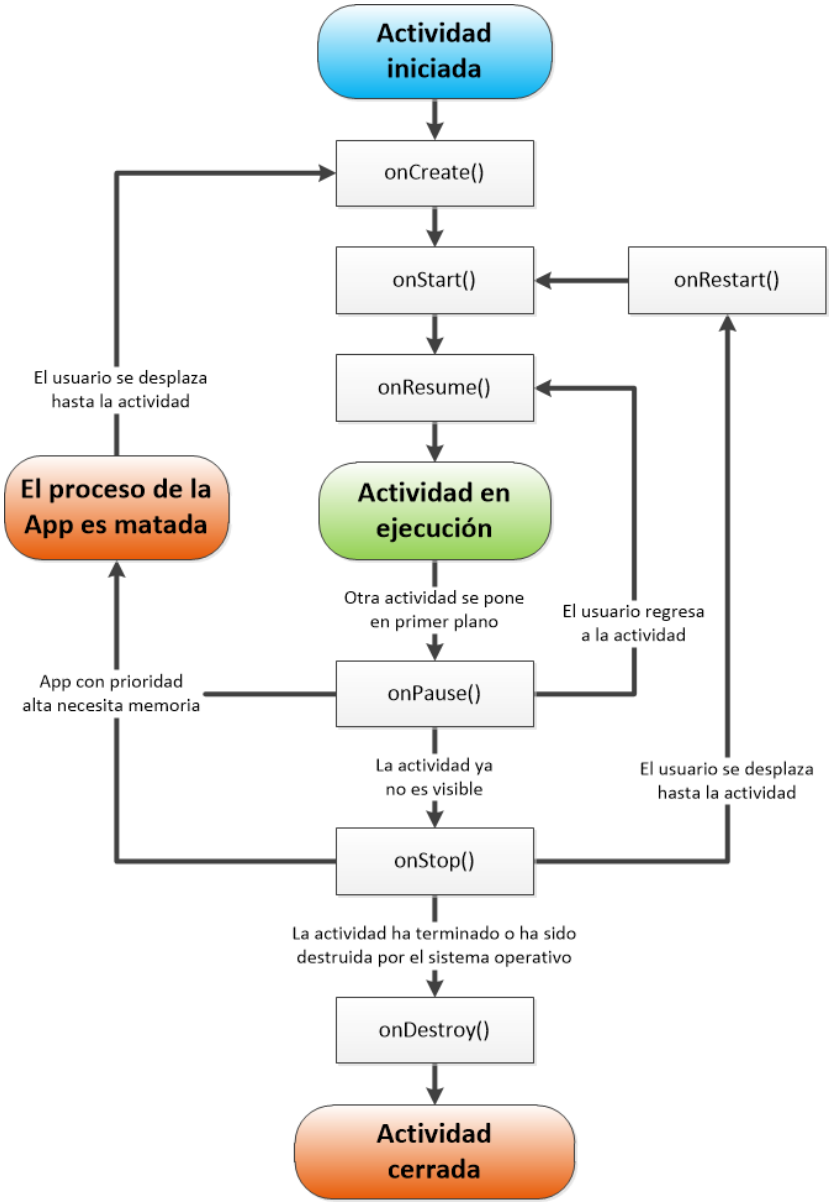


Fig.9. Ciclo de vida de una Actividad [9].

Para la creación de un proyecto en Android Studio simplemente se tienen que seguir los pasos indicados bajo la opción “Create New Project”. Solamente se ha de tener en consideración el SDK mínimo con el que se va programar. Las versiones más altas son más potentes y tienen más recursos pero las aplicaciones resultantes solo podrán ejecutarse en terminales más modernos. Android Studio informa en el momento de la creación de qué porcentaje de terminales, conectados al Google Play Store podrán ejecutar la aplicación desarrollada. Para ese proyecto se ha optado por el DSK 24, siendo la aplicación desarrollada válida para más del 90% de los dispositivos actuales.

Una vez elegido el nombre de la actividad principal, se genera toda la estructura de carpetas del proyecto y se puede comenzar con la programación propiamente dicha. En la zona de diseño se podrá acceder a los ficheros .xml de la carpeta Layout, para diseñar el interfaz de usuario de manera gráfica (Design) o trabajar directamente con el fichero .xml en texto (Text) (Fig.10). Así mismo, se podrán abrir los ficheros java mediante un editor con múltiples facilidades, como completado automático, corrección de errores léxicos, etc.

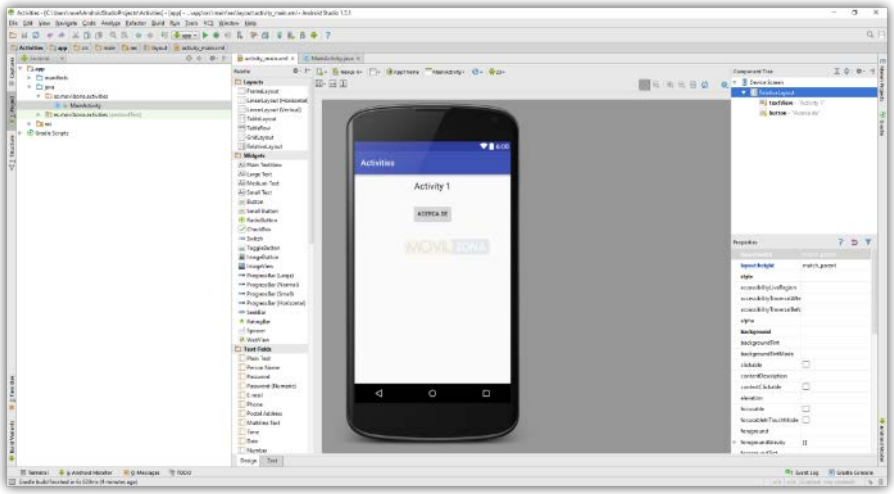


Fig.10. Interfaz principal de Android Studio.

Una vez implementada la programación, Android Studio ofrece un emulador para la depuración de la aplicación desarrollada. Sin embargo, esta opción no es soportada en el PC utilizado para el desarrollo del TFG, debido a su microprocesador (AMD A8). Es por ello que se ha utilizado depuración en dispositivo físico mediante conexión USB. Para ello, es necesario, por un lado, instalar el Driver USB (esto se puede hacer desde la herramienta de SDK Manager) del Android Studio (Fig. 11).

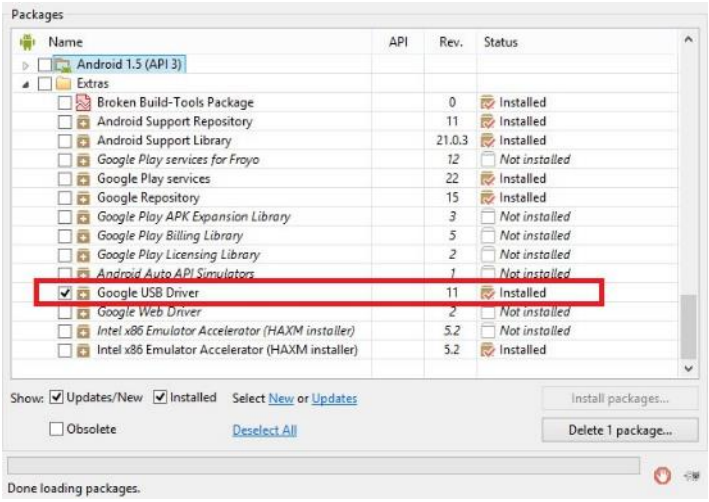


Fig.11. Instalación del Google USB Driver.

A continuación, desde el Administrador de Dispositivos del S.O. (Windows en este caso) debe seleccionarse el dispositivo (una vez conectado mediante USB) y seleccionar “actualizar controlador” con la dirección de éste. Si el controlador se ha descargado desde Android Studio este se encontrará en la carpeta “…\AppData\Local\Android\sdk\extras\google\usb_driver”.

Si no se puede conseguir instalar este driver, como ocurrió en el caso de este TFG, también se puede intentar actualizar mediante la lista de controladores de dispositivo del equipo, eligiendo, para equipos Android, un controlador de ADB interface.

Por otro lado, será necesario activar la función de depuración por USB en el dispositivo físico que se desee utilizar. Para ello se debe llegar a la opción de Depuración USB navegando a través de los menús Configuración -> Aplicaciones -> Desarrollo -> Depuración USB. En Android 4.2 y posteriores, esta opción está oculta, con lo que para habilitarla habrá que acceder a Ajustes -> Información del Teléfono y pulsar repetidas veces sobre la opción Número de compilación hasta que aparezca un mensaje indicando que ya se poseen capacidades de desarrollador, quedando liberada la opción de depuración USB mencionada con anterioridad.

Una vez realizadas estas opciones, al seleccionar la ejecución del nuestro proyecto en Android Studio nos aparecerá como una opción nuestro terminal físico (Fig.12).

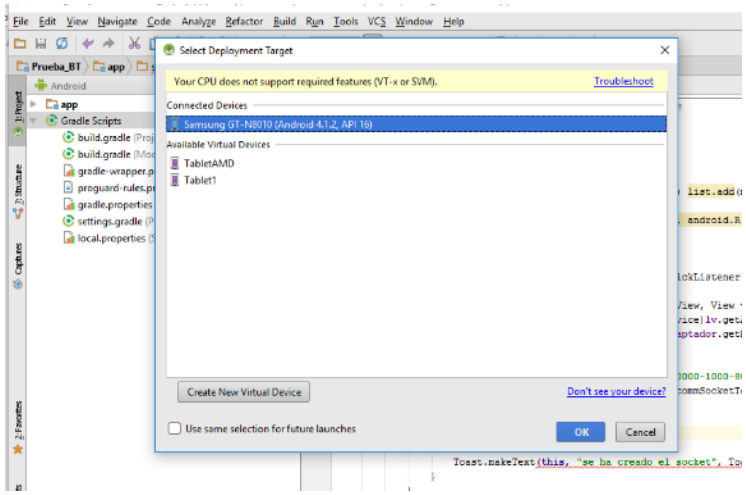


Fig.12. Depuración con dispositivo físico.

Para finalizar esta breve introducción a Android Studio y los puntos más interesantes de su instalación, se debe comentar que, además de las librerías básicas incluidas con la descarga del Android Studio, la comunidad de desarrolladores ofrece innumerables librerías (o dependencias, como son

denominadas) gratuitas para casi cualquier necesidad de programación. La inclusión de cualquier librería es muy sencilla, basta con seleccionar en el menú Archivo - > Estructura del proyecto -> Dependencias -> Añadir -> Dependencia de archivos y agregar el fichero .jar de la librería. Esto hará dos cosas. Por un lado, incluirá la librería en la estructura de ficheros del programa. Por otro actualizará el build.gradle incluyendo la dependencia (compile files ('nombreLibreria.jar')), pudiéndose desde este momento utilizar todas las clases y métodos de la misma.

1.3.5. Electrocardiograma (ECG) y Electromiograma (EMG)

Para concluir esta breve introducción de conceptos utilizados en el TFG se van a describir las principales características de las dos señales tratadas: el electrocardiograma (ECG) y el electromiograma (EMG) [10].

Electrocardiograma ECG

El electrocardiograma es la representación gráfica de la actividad eléctrica del corazón. Como cualquier otro músculo, el cardiaco se contrae debido a una señal eléctrica que se propaga a través de sus fibras nerviosas (despolarización). Esta despolarización, consistente en un cambio de la carga en la membrana de la fibra, se origina en el denominado nódulo sinusal o sinoauricular (SA) en la parte superior de la aurícula derecha (Fig. 13). La señal se propaga a través de las aurículas contrayéndolas hasta llegar al nódulo auriculoventricular (AV), donde la señal se retarda un poco y luego progresa de nuevo hacia los ventrículos, que pasan a contraerse. A medida que la despolarización va progresando y una vez concluida la contracción, las fibras se van repolarizando regresando a su estado de carga normal.

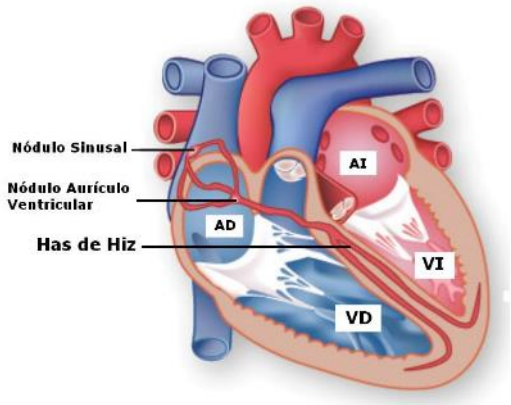


Fig.13. Anatomía del corazón humano (Fuente Infosalud)

Para medir esta señal, se utilizan una serie de electrodos colocados sobre la piel. Es importante tener en cuenta que, en función de la posición de los electrodos, la señal del electrocardiograma puede variar, pudiéndose incluso no captarse (si la señal se desplaza perpendicularmente a los electrodos). Para recoger con total fidelidad el desplazamiento del vector de polarización en el corazón, se utilizan diversas posiciones de los electrodos, para analizar el ECG en diferentes direcciones. Esto es lo que se conoce como derivaciones. Existen doce derivaciones principales (Fig. 14): tres Bipolares (I,II y II) medidas entre dos electrodos; y nueve Unipolares, medidas respecto a un nodo de referencia (generalmente, la media obtenida en las medidas de otros electrodos), divididas a su vez éstas entre las aumentadas (aVR,aVL,aVF) y las precordiales (de la V1 a la V6),.

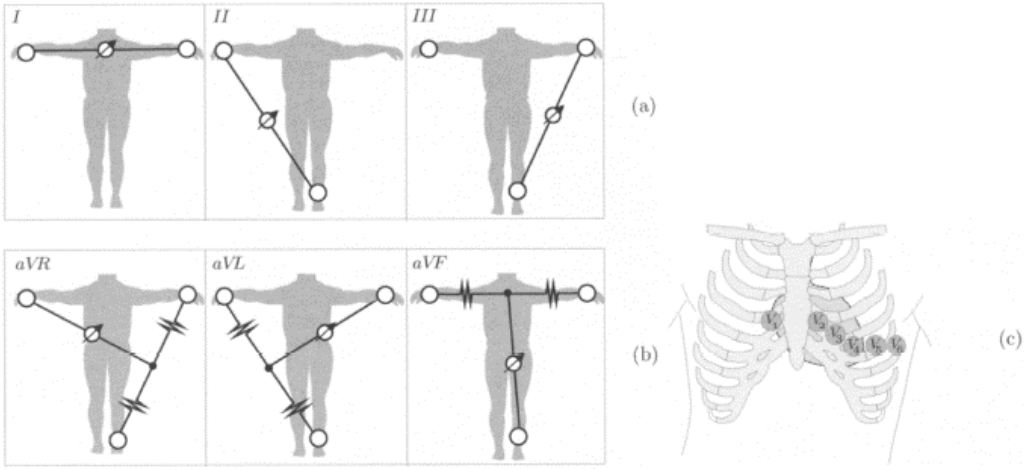


Fig.14. Doce derivaciones del ECG. Bipolares (a), Aumentadas (b) y Precordiales (c).

La señal obtenida, u onda del ECG tiene la forma presentada en la Fig.15. Se distinguen dos mínimos (Q-S) y tres máximos (P-R-T). La onda P se corresponde a la despolarización auricular y comienzo del latido. El complejo QRS a la despolarización de los ventrículos y la onda T a la repolarización de estos. La onda correspondiente a la repolarización de las aurículas queda oculta por el complejo QRS.

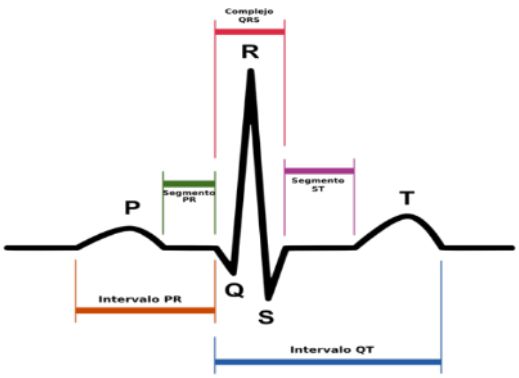


Fig.15. Onda del ECG.

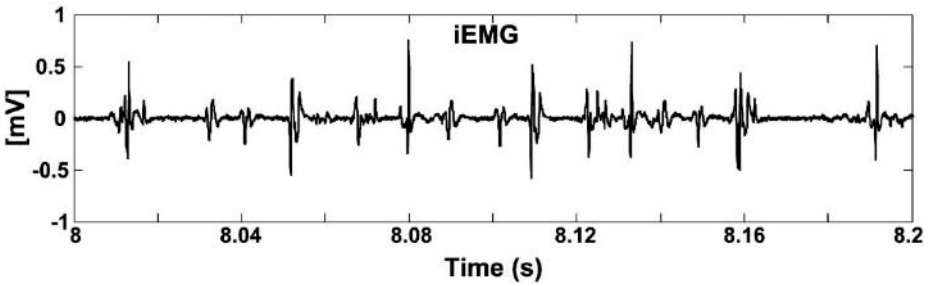
Al intervalo entre dos ondas R se le denomina ritmo cardiaco. Hay que tener en cuenta que la señal del ECG se puede ver alterada por muchos factores, siendo algunos naturales y otros interferencias no deseadas. Entre los primeros se encuentra el estado anímico, la actividad física y el ritmo respiratorio, que pueden aumentar o disminuir el ritmo cardiaco. Entre los segundos, son especialmente molestos los ruidos de interferencia electromagnética por alimentación o equipos cercanos (que pueden ser paliados por un correcto acondicionamiento de la señal) e interferencias electromiográficas, producidas por los músculos toráxicos, que también producen potenciales de acción que pueden ser captados por los electrodos. Para disminuir este efecto, se busca la mayor inmovilización del paciente, ya que resulta muy difícil de eliminar, al poseer componentes frecuenciales similares a las del complejo QRS.

Electromiograma EMG

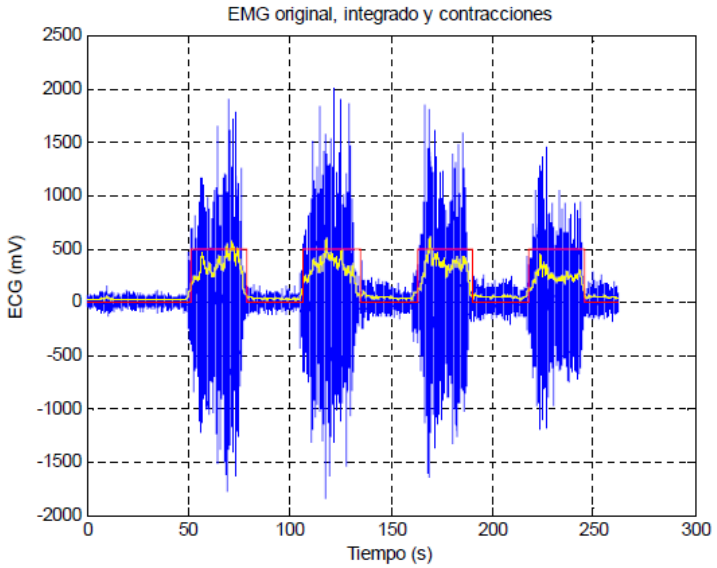
La señal de electromiografía tiene la misma base que el ECG. Se trata de obtener, mediante electrodos, la información del potencial de acción que activa una unidad motora. Cuando se quiere flexionar un músculo, las neuronas motoras activan un conjunto de fibras musculares. Para una mayor fuerza, o bien se activan más fibras musculares (reclutamiento espacial) o se incrementa la cantidad de disparos de potenciales por unidad de tiempo (reclutamiento temporal).

En la obtención del EMG se utiliza un electrodo de referencia situado lejos de los paquetes musculares (codo, rodilla) y dos o más electrodos sensores. Estos pueden colocarse de forma intramuscular (técnica invasiva mediante la colocación de los electrodos en el interior del músculo) o superficial (con electrodos adheridos a la piel). En la primera, se pueden distinguir los disparos de las diferentes unidades motoras, generalmente a un ritmo constante, y el aumento de este ritmo o el reclutamiento de nuevas unidades motoras si es necesario (Fig.16a). Estas señales suelen tener componentes frecuenciales altas de interés (varios KHz), por lo que hay que tener cuidado en su filtrado. Por otro lado, en los EMG superficiales, la señal no suele tener mucha precisión al recogerse señal de muchas unidades motoras, las cuales se superponen, siendo muy complejo analizar el reclutamiento espacial y/o temporal. No obstante, se puede utilizar fácilmente para detectar el inicio del impulso y el nivel de éste. Por ello, en general, la señal suele ser filtrada y rectificada para obtener una integración de todas las activaciones de unidades motoras durante el movimiento (Fig.16b).

Las principales fuentes de ruido son las de la alimentación o movimientos indeseados del electrodo, aunque estos suelen ser de baja frecuencia y pueden ser fácilmente filtrados en el acondicionamiento de señal.



(a)



(b)

Fig.16. Señal del EMG intramuscular (a) y EMG de superficie (b) con la señal original (azul) y una vez procesada (amarillo).

2

SISTEMA DE CAPTURA DE SEÑALES BIOMÉTRICAS (ARDUINO)

Este segundo capítulo contiene la descripción del sistema necesario para la captura de las señales biométricas (ECG y EMG). Esta captura se realiza mediante un sistema basado en Arduino más la extensión e-Health. Se detallan las conexiones físicas necesarias, así como la programación necesaria para su correcto funcionamiento, la cual puede encontrarse igualmente en el CD adjuntado.

2.1.- Estructura Básica

Como ha sido mencionado, el sistema está basado en un Arduino más la extensión e_Health, que permitirá la adquisición y acondicionamiento de las señales de ECG y EMG.

El objetivo de esta primera fase es obtener las señales biológicas en la conexión serie proporcionada por Arduino a través del conector USB con el PC, de tal manera que, posteriormente se puedan usar estos datos para la fase del desarrollo de la aplicación Android. Ambas medidas deben ser integradas en un solo sketch. Por ello se ha de permitir la selección de la medida del ECG o el EMG mediante el nivel de una señal digital de entrada.

2.2.- Estructura Hw

El e-Health incluye el sistema de acondicionamiento de las señales de ECG y EMG, el cual es importante conocer, pues permitirá eliminar parte del procesado *software* de las mismas [11].

El EMG se obtiene mediante el uso de tres electrodos, dos de señal y un tercero que se usa como referencia, siendo la señal obtenida una medida diferencial entre los electrodos positivo y negativo. La colocación de éstos será en el pecho del usuario, según la Fig. 17. Al ser una medida diferencial, en sentido estricto no se corresponde con ninguna derivación de las clásicas.

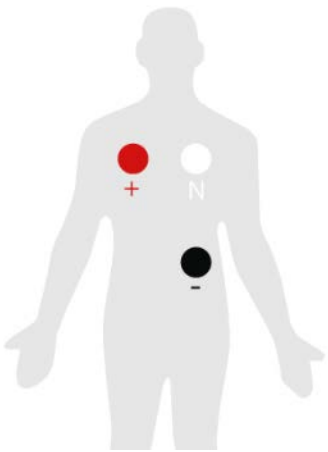


Fig.17. Posicionamiento de los electrodos para ECG [5]

En el circuito de acondicionamiento (Fig.18) se pueden ver dos bloques diferenciados. Por un lado, se obtiene la señal de referencia mediante un seguidor de tensión, mientras que, por otro lado, se utiliza el amplificador de instrumentación INA321EA de Texas Instruments, con una ganancia por defecto de 5. Las entradas diferenciales están referidas a VR (electrodo de

referencia) siendo $(\mathrm{EGC}_+ - \mathrm{EGC}_+)\frac{\mathrm{R}_1}{\mathrm{R}_1+\mathrm{R}_4} \approx (\mathrm{EGC}_+ - \mathrm{EGC}\_)4.96$, así pues, la salida del INA321EA es, básicamente, cinco veces la diferencia entre el terminal positivo y el negativo. El electrodo de referencia está conectado a la salida de un amplificador operacional en estructura integradora, que permite proteger al paciente de posibles sobretensiones.

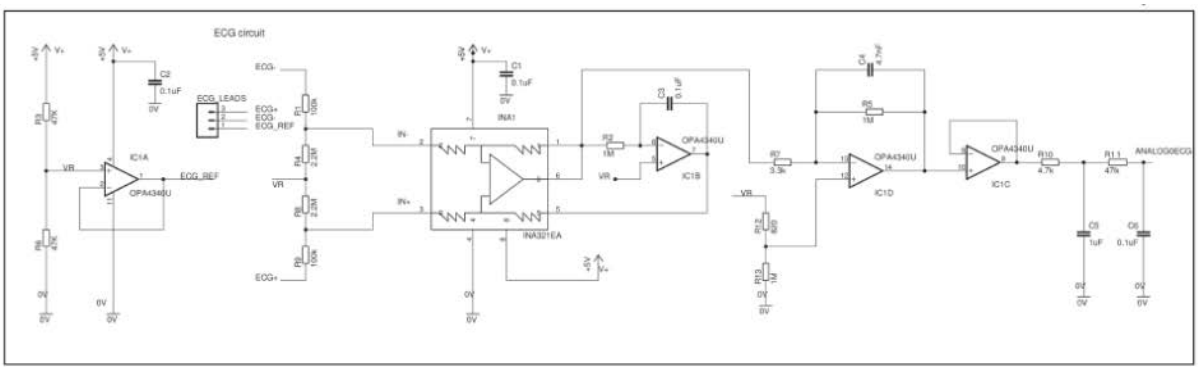


Fig.18. Circuito de acondicionamiento del ECG

El sistema continúa llevando la señal diferencial amplificada a un nuevo amplificador, ahora en configuración inversora, de ganancia $\mathrm{G} = -\frac{\mathrm{R}_5}{\mathrm{R}_7} \approx -303$. El comportamiento es paso bajo gracias a C4 (frecuencia de corte, 33.86 Hz).

Finalmente se concluye con un nuevo seguidor de tensión que permite atacar dos etapas simples RC de filtrado paso bajo de frecuencia de corte 33.9 Hz, uno con capacidad de 1μF y otro de 0.1μF, probablemente intentando que funcione a frecuencias más altas elevando la frecuencia de resonancia de la capacidad real.

Hay que recordar que la combinación de los diferentes filtros paso bajo hará disminuir la frecuencia de corte total, ésta se encuentra aproximadamente (considerando los tres filtros independientes, aunque esto no es del todo cierto), a $\mathrm{f}_{0\mathrm{total}} = \mathrm{f}_0\sqrt{2^{1/3}-1}$ =17.3 Hz. No obstante, ésta sigue siendo lo suficientemente alta como para poder obtener la señal cardiaca a máximo rendimiento (180 pulsaciones por minuto, 3 Hz), con suficiente orden de armónicos como para poder analizar incluso tiempos de subida y bajada de dicha señal en el complejo QRS.

Por su parte, el sistema de acondicionamiento para el sensor del EMG consta de cuatro etapas (Fig.19): Una amplificación diferencial, un rectificado, un filtrado activo y una etapa de amplificación regulable.

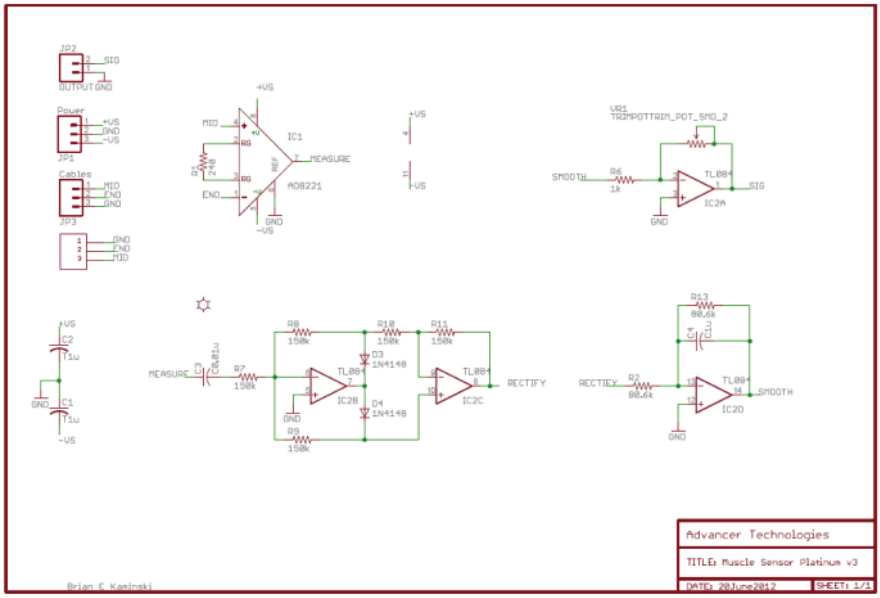


Fig.19. Circuito de acondicionamiento del EMG

La primera etapa hace uso del AD8221 de Analog Device para amplificar la señal diferencial MID-END. La ganancia es regulable mediante R1, según la relación $G = 1 + \frac{49{,}4\ K}{R_1} \approx 207$

La siguiente etapa realiza el rectificado de la señal. Se tiene en primer lugar un filtrado paso alto a la frecuencia de 106 Hz, que elimina señales de baja frecuencia. Después el rectificador propiamente dicho. En función de que la señal de entrada sea positiva o negativa, conducirá un diodo u otro, atacando a la segunda etapa por la entrada positiva o negativa y, por tanto, invirtiéndose o no dicha señal, con el efecto final de tener la señal rectificada a la salida.

La tercera etapa es un inversor, con comportamiento paso bajo y ganancia unidad. Su frecuencia de corte es de 2Hz, que elimina todas las componentes de alta frecuencia que aparecen del rectificador. De esta forma se obtiene la envolvente de la señal, que agrupa la información de todas las fibras participantes en el movimiento muscular.

Finalmente se tiene un amplificador inversor, cuya ganancia se puede regular con el potenciómetro RV1.

Los electrodos han de posicionarse en función del músculo que se desee analizar. En los experimentos realizados en el presente TFG se ha analizado el bíceps del brazo, con lo cual, los electrodos han sido posicionados en las posiciones indicadas en la Fig.20.

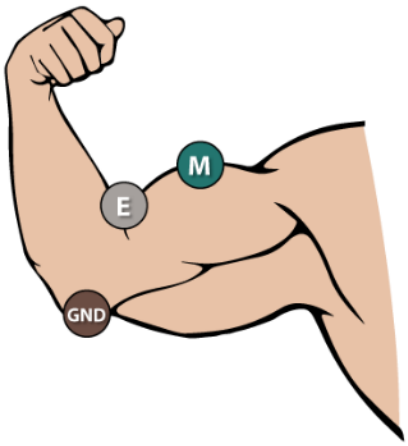


Fig.20. Posicionamiento de los electrodos para el EMG [5]

Una vez analizada la electrónica de acondicionamiento de señal, los pasos necesarios para la conexión física del sistema son los siguientes: Conexión de la extensión e-Health al Arduino ONE, conexión del Arduino ONE al PC, conexión de los electrodos de medida y preparación de un latiguillo para la selección de la bioseñal a capturar a la extensión e-Health. Hay que tener en cuenta que, además, se tendrá que colocar el puente incluido en el e-Health en la posición correspondiente a la medida deseada. Todas estas conexiones se muestran en la Fig.21.

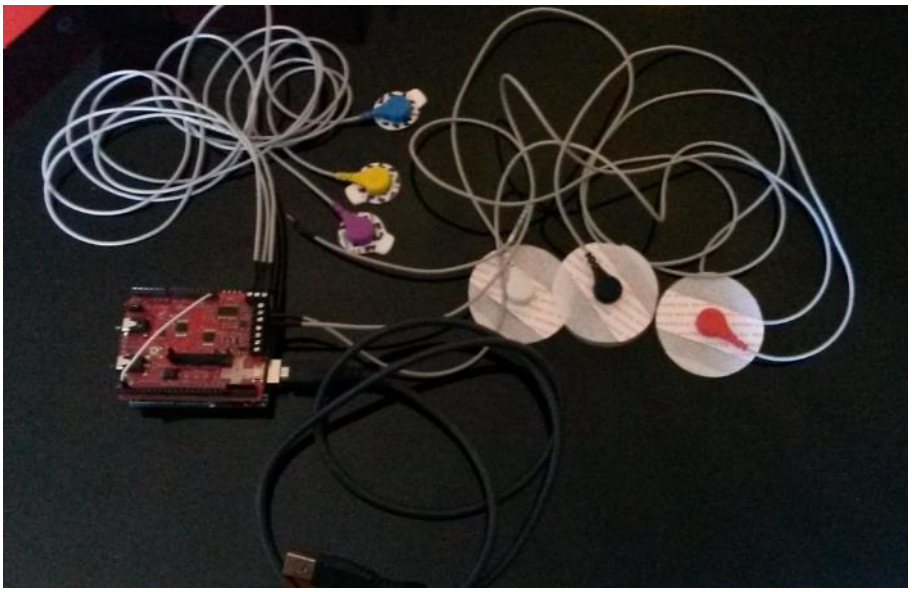


Fig.21. Sistema Hw (Arduino,e-Health y electrodos)

2.2.- Estructura Sw

La programación del Arduino para la captura del ECG y del EMG es muy sencilla gracias a la librería suministrada por el propio e-Health (“eHealth” y “PinChangeInt”, aunque este último no es imprescindible para las medidas de ECG y EMG). Es importante tener en cuenta que estas librerías solo se pueden utilizar con el IDE Software Arduino 1.0.1, por lo que la programación con la última versión del IDE de Arduino no funciona correctamente.

El sketch es muy sencillo. El bloque de declaración incluye la librería de e-Health e inicializa un contador para limitar el número de muestras obtenidas. Hay que recordar que el objetivo de esta primera fase es obtener un fichero con muestras de las señales biológicas por lo que no es necesario que se esté indefinidamente tomando datos. Finalmente se crea una variable para indicar que el pin seleccionador de la medida deseada será el “2”

```
#include <eHealth.h>
int cont=0;
int pinSeleccion=2;
```

El bloque setup() iniciará la velocidad del puerto serie a 115200 baudios, e indicará que el pin 2 será un pin de entrada.

```
void setup() {
                Serial.begin(115200);
                pinMode(pinSeleccion,INPUT);
                //Pin 2 HIGH, medida del ECG
                //Pin 2 LOW, medida EMG
}
```

Finalmente, el bloque loop() capturará continuamente la señal deseada, bien el ECG o bien el EMG en función del nivel encontrado en el pin digital 2 del Arduino (nivel alto para ECG, nivel bajo para EMG). No hay que olvidar que también hay que cambiar físicamente el puente ECG/EMG de la extensión para seleccionar la medida correcta. Una vez tomadas 1000 muestras, se deja de capturar. Cada dato capturado es enviado por el puerto serie con tres decimales y se separa con un fin de línea y retorno de carro. Finalmente se espera 10 milisegundos antes de repetir el bucle. Por tanto, se tiene una frecuencia de muestreo de 100 Hz, suficiente para las señales tratadas.

```
void loop() {
        if (digitalRead(pinSeleccion)==LOW){
                if (cont<1000){
                                delay(10);
                                float ECG =eHealth.getECG();
                                Serial.print(ECG, 3);
```

```
                    Serial.println("");
                    cont++;}
        }
else {
        if (cont<1000){
                    delay(10);
                    float EMG =eHealth.getEMG();
                    Serial.print(EMG, 3);
                    Serial.println("");
                    cont++;}
        }
}
```

Los datos así capturados pueden ser mostrados en el terminal del puerto serie y ser fácilmente trasladados a un fichero de texto para su posterior tratamiento con la aplicación Android, tal y como se expondrá en el siguiente capítulo.

2.3.- Pruebas y verificación

Para el sistema Arduino de captura de las señales biológicas el proceso de diseño y pruebas siguió el siguiente esquema:

- Prueba del Arduino ONE: conexión y ejecución del sketch de prueba “Blink”
- Prueba del puerto serie: Envío de la frase “Prueba del puerto Serie” con diferentes velocidades en baudios.
- Selección de medida: Envío por el puerto serie de la palabra “ECG” o “EMG” en función del nivel impuesto en el pin 2.
- Prueba de interconexión e-Health/Arduino ONE: Conexión de la extensión e-Health y repetición de las pruebas anteriores.
- Prueba e-Health: Captura de los datos de ECG y el EMG y envío al monitor serie.

Los resultados demostraron que el sistema funciona correctamente a cualquier velocidad de transmisión en baudios, aunque los primeros caracteres capturados (tres o cuatro) pueden ser erróneos a veces. Se cree que el problema es del arranque del monitor serie que no coge los primeros datos enteros. No obstante, y en cualquier caso, como el sistema final este problema no existe.

3

DESARROLLO DE LA APLICACIÓN MÓVIL

Este tercer capítulo contiene la descripción de la aplicación móvil desarrollada mediante Android Studio, la cual constituye en núcleo del presente TFG. Su misión es la de guiar al usuario entre las distintas opciones disponibles, referidas tanto a las señales ECG como EMG, realizando su representación, almacenamiento y recuperación, así como un par de pequeños experimentos relacionados. Lógicamente, también tiene el objetivo de representar gráficamente dichas señales o los resultados de los experimentos.

La estructura general del *software*, así como el detalle de cada una de sus funcionalidades se describe con detalle, así como el interfaz de usuario. Por motivos de espacio no se incluye el código completo, el cual se anexa en CD (proyecto TFG_ECG_EMG), sino que se presta especial atención a aquellas partes que son de relevancia o han necesitado mayor trabajo de desarrollo.

3.1.- Estructura Básica

La aplicación está compuesta por ocho ventanas o actividades (Activity) cada una de las cuales se encarga de una función determinada:

Activity main: Se trata de la actividad de acceso a la aplicación. Contiene los botones que permiten acceder a las funciones relacionadas con el electrocardiograma o con el electromiograma, así como a una pequeña referencia de ayuda o manual de usuario.

Activity Cardio: Permite seleccionar entre la opción de "Graficos" o de "Laboratorio" para el ECG, así como retornar a la actividad principal.

Activity CardioGraficos: Permite seleccionar, y representar en un gráfico, el ECG almacenado en el fichero de texto. Esta representación puede detenerse para analizarla detalladamente y reanudarse cuando el usuario lo desee. En la última fase (capítulo 5) del TFG también permitirá representar una ECG capturado por el e-Health y el sistema Arduino y enviado por Bluetooth, así como almacenar en un fichero esta señal.

Activity CardioLab: Permite representar en un gráfico el ECG, así como obtener el ritmo cardiaco a partir de él. De la misma manera, permite reproducir dos tipos de música para ver como esta afecta a dicho ritmo. En esta fase, antes de obtener las señales reales a tiempo real, se usarán las señales pregrabadas en archivos de texto para comprobar su funcionalidad.

Activity Miograma: Permite seleccionar entre la opción de "Gráficos" o de "Laboratorio" para el EMG de manera similar a la actividad Cardio.

Activity MioGrafico: Funciona de manera similar a la actividad CardioGraficos. Permite representar en un gráfico los EMG almacenados en ficheros de texto y se preparará para la representación de señales captadas por el sistema Arduino en la última fase del TFG.

Activity MioLab: Permite representar en un gráfico el EMG, así como la intensidad y la duración del último impulso nervioso enviado

al musculo. Al igual que en la actividad CardioLab se utiliza en esta fase señales almacenadas en un fichero.

Activity Acercade: Muestra información básica de la aplicación y su uso.

El organigrama de acceso a las distintas actividades se muestra en la Fig. 22, representado mediante los layouts de las actividades.

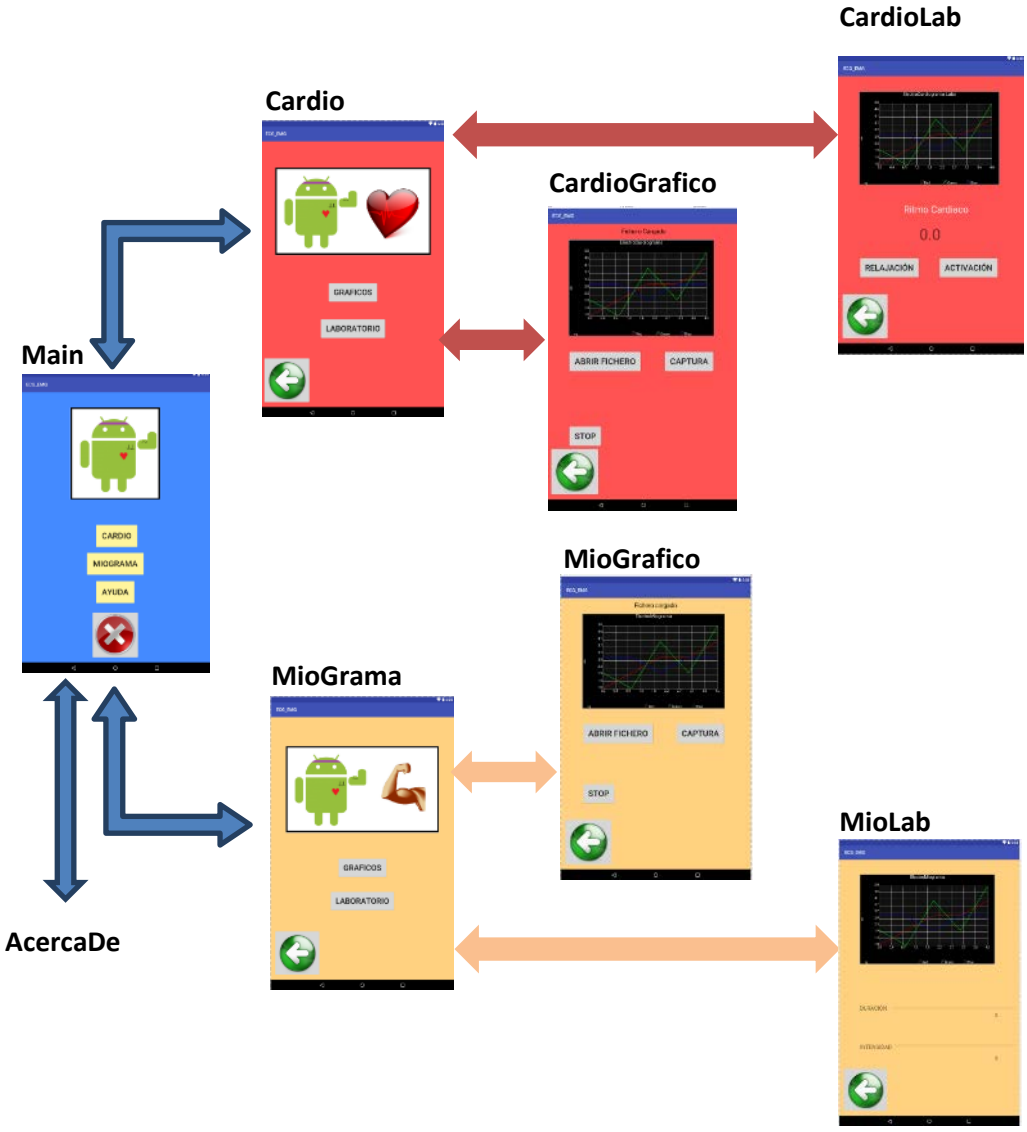


Fig. 22. Diagrama de actividades de la aplicación Android

3.2.- Electrocardiografía

La actividad Cardio (Fig. 23) presenta tres botones que permiten acceder tanto a la actividad CardioGraficos de representación y captura (Gráfico), como a la actividad CardioLab para el experimento (Laboratorio), así como retroceder a la actividad Main (icono de la flecha).

Fig. 23. Layout de la actividad Cardio

3.2.1. - Representación

Implementada en la actividad CardioGraficos, permite la representación gráfica dinámica del ECG almacenado en un fichero de texto (Fig.24).

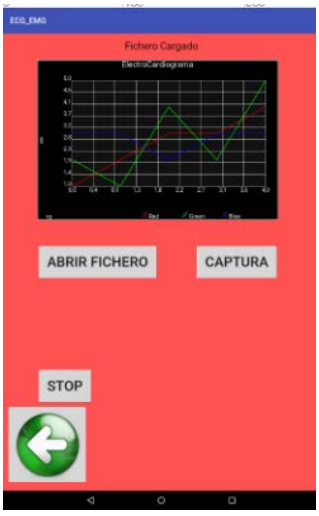


Fig. 24. Layout de la actividad Cardiograficos

Una vez la representación se encuentre en ejecución, ésta se podrá parar o reanudar, así como cargar un nuevo fichero. Se incluirá igualmente un botón para acceder a la captura en tiempo real del ECG cuando se integre la comunicación Bluetooth, aunque en esta fase del proyecto no tendrá efecto como se verá más adelante. El diagrama de flujo de esta actividad se muestra en la Fig. 25.

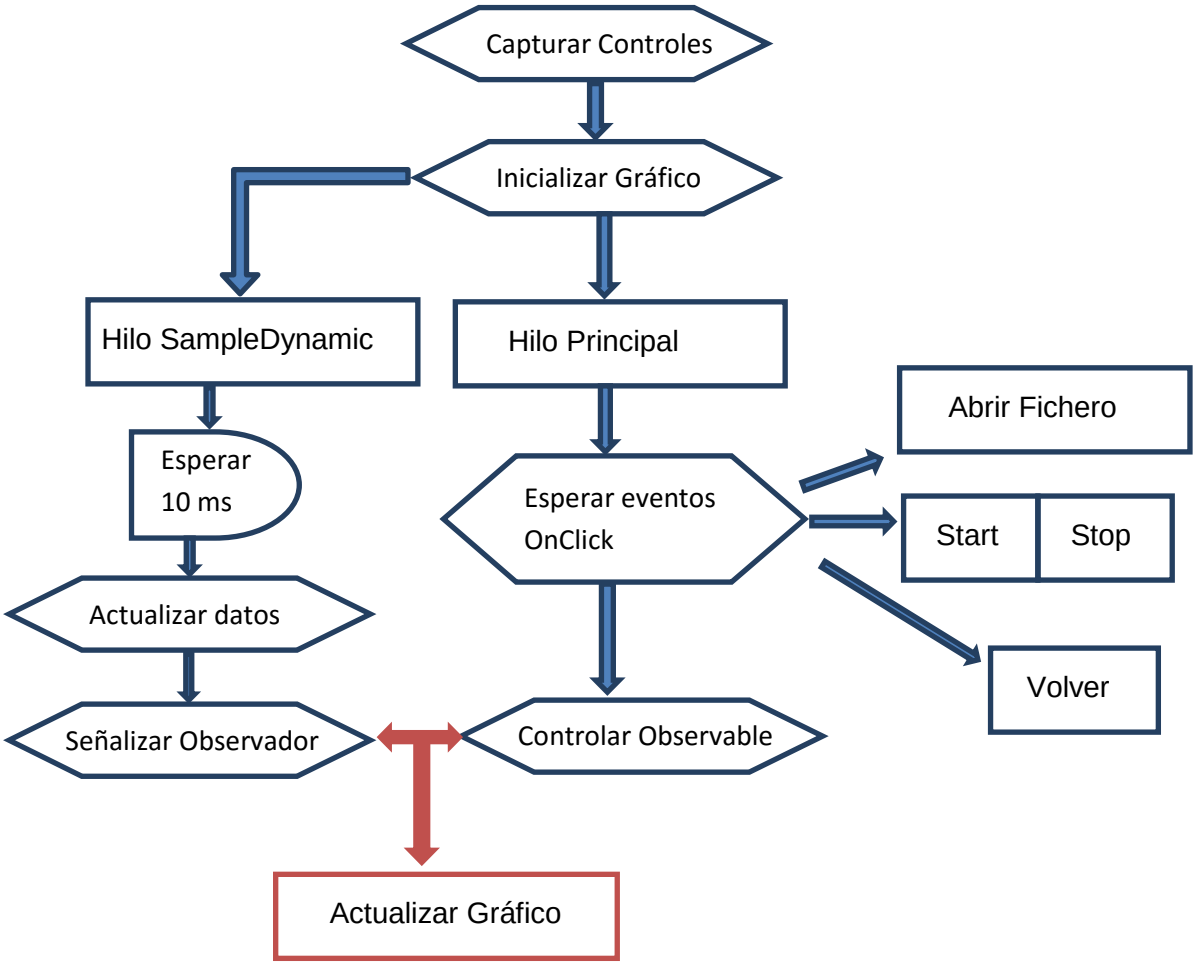


Fig. 25. Diagrama de flujo simplificado de la actividad Cardiograficos

La actividad está basada en la librería o dependencia AndroidPlot [12], la cual incluye todas las clases y métodos que permite la representación dinámica en un gráfico actualizable (Objeto XYPlot) de cualquier conjunto de datos dinámicos obtenidos en la ejecución en paralelo de un hilo (Threat) con el hilo principal.

La comunicación entre el hilo principal y la actualización dinámica de los datos sigue un modelo Observador-Observable. De esta manera, cada vez que los datos se actualicen en el hilo correspondiente, se indicará al hilo principal para que actualice el gráfico.

De esta forma, la actividad realiza lo siguiente:

Hilo Principal

- En primer lugar, se llama a un método creado que recoge todos los controles del layout que pueden ser utilizados por el usuario.
- Se implementa el Observador, para que, cuando los datos se actualicen se actualice el gráfico mediante el método redraw().

```
private class MyPlotUpdater implements Observer {
    Plot plot;

    public MyPlotUpdater(Plot plot) {
        this.plot = plot;
    }

    @Override
    public void update(Observable o, Object arg) {
        plot.redraw();
    }
}
```

- Se crea el hilo en paralelo (clase “runnable”) mediante un objeto SampleDynamicXYDatasource y se vincula al Observador. Los datos iniciales consisten en un vector inicializado a valores nulos, por lo que se representará una línea plana de cero Voltios hasta que no se cargue algún fichero.

```
for(int k=0;k<2000;k++){datos.add((float)0);}

data = new SampleDynamicXYDatasource();
SampleDynamicSeries ECGSeries = new SampleDynamicSeries(data,
"ECG");

// Vinculación del observador a los datos
data.addObserver(plotUpdater);
```

- Se da formato al gráfico. Se decide dibujar en línea continua con pequeños símbolos en las muestras reales obtenidas y los ejes se fijan a los valores esperados (menos de 4 Voltios).

El hilo principal queda así a la espera de que se produzcan eventos onClick (pulsación) en los controles o que el Observable en el hilo del objeto SampleDynamicXYDatasource indique que hay nuevos datos a representar.

Hilo SampleDynamic

- Su método run{} se ejecuta continuamente mientras no se pulse el botón del stop. Básicamente incrementa una posición (variable global “*paso*”) la lectura del vector de datos y se notifica al Observador que reactualizará el gráfico. Este incremento se realiza cada 10

milisegundos, correspondiente al intervalo entre dos muestras del ECG obtenidas con el Arduino. Como control del desbordamiento en la lectura del vector de datos, la variable *"paso"* se reinicializa a cero si el paso actual más el tamaño de representación supera el valor de la posición final del vector de datos (no hay suficientes datos para hacer una representación. El tamaño de representación ha sido elegido de 500 muestras, correspondiente a 5 segundos (entre 5 y 10 latidos)

```
public void run() {
    try {
        keepRunning = true;
        while (keepRunning) {

            Thread.sleep(10); // Actualización cada 10 msg
            paso++;
            if (paso==datos.size()-SAMPLE_SIZE){paso=0;}
            notifier.notifyObservers();
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

- Los datos para representar serán obtenidos por el objeto XYPlot mediante llamadas a distintos métodos de esta clase, por lo que es necesario reescribirlos para que hagan lo deseado. Como se ha indicado, para el eje Y se utilizaran los valores almacenados en el vector *"datos"* desde la posición *"paso"*. Por su lado, el eje X originalmente solo usa un índice de cero al valor máximo de muestras representadas menos uno. Para que se corresponda con el índice temporal, se multiplica por 0.01, obteniendo el eje en tiempo en milisegundos.

```
public Number getX(int index) {
    if (index >= SAMPLE_SIZE) {
        throw new IllegalArgumentException();
    }
    return index*0.01;
}

public Number getY(int index) {
    if (index >= SAMPLE_SIZE) {
        throw new IllegalArgumentException();
    }
    float amp=datos.get(index+paso);
    return amp;
}
```

Métodos OnClick

La actividad posee varios controles que pueden ser usados por el usuario, cada uno de ellos asociados a una respuesta:

- Abrir Fichero
- Start
- Stop
- Lista de Ficheros
- Capturar
- Volver

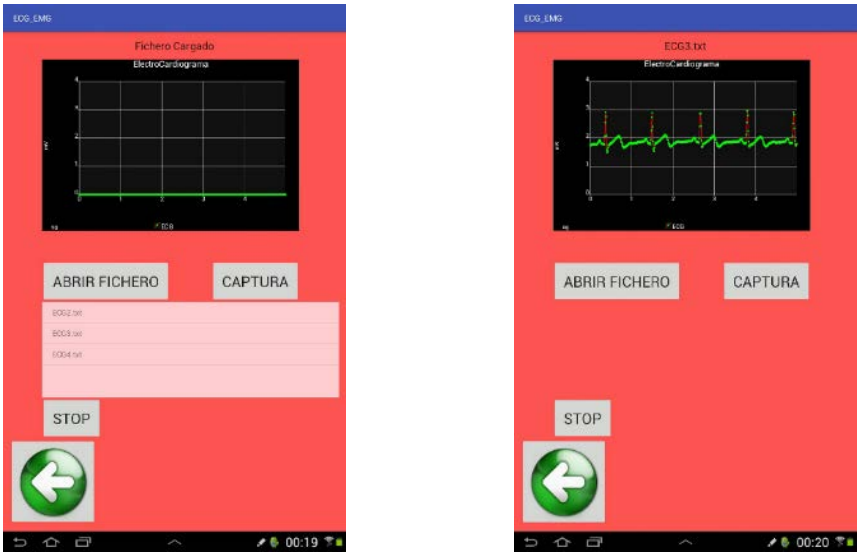
Stop y **Start** simplemente pausan o reanudan el hilo de actualización de los datos. Señalar que estos controles nunca están activos simultáneamente, sino que su atributo de visibilidad se alterna entre Visible e Invisible respectivamente. De esta forma se evita que se reanude la actualización de datos cuando ya está activa (produciría el lanzamiento de otro nuevo hilo de actualización, obteniéndose el doble de frecuencia de actualización).

```
public void onClickStop(View v) {
    onPause();
    boton_start.setVisibility(View.VISIBLE);
    boton_stop.setVisibility(View.INVISIBLE);
}

public void onClickStart(View v) {
    boton_start.setVisibility(View.INVISIBLE);
    boton_stop.setVisibility(View.VISIBLE);
    onResume();
}
```

Abrir Fichero permite accede a un fichero de texto con los datos de una captura de ECG y realizar su representación en el objeto XYPlot. Estos ficheros se encuentran almacenados en la memoria externa no extraíble del dispositivo. Se ha creado una variable tipo String para almacenar el camino de almacenamiento:

```
private String nomDirArch="/storage/sdcard0/IngSalud/ECG/";
```

Se cargan en una lista todos los ficheros localizados en dicho camino y se cargan en un control tipo ListView, que se encontraba invisible hasta el momento (Fig.26a). El usuario podrá entonces seleccionar el archivo que desee cargan presionando en él, lo que producirá un nuevo evento onClick que realizará la lectura del fichero seleccionado. Desde ese momento, el objeto XYPlot representará los datos del archivo (Fig.26b) mientras que en un control tipo TextView sobre el gráfico se mostrará el nombre del fichero.



(a) (b)

Fig. 26. Carga de ficheros (a) y representación de un fichero cargado (b).

La lectura del archivo se ha realizado en un método auxiliar que realiza una lectura clásica de archivo de texto línea a línea mediante la clase BufferReader, convirtiendo las cadenas extraídas en variables tipo float. Señalar que, mientras se realiza la carga, la actualización de datos entra en pausa, reanudándose una vez cargado el archivo. De esta manera se evita que la representación intente acceder al vector de datos mientras este se está modificando. Así mismo, la variable *"paso"* que indica la posición actual de lectura del vector se reinicializa a cero. Finalmente, se vuelve a ocultar el control ListView con la lista de archivos disponibles.

```
public void onClickBuscarFich(View v) {
    item = new ArrayList<String>();

    File f = new File(nomDirArch);
    File[] files = f.listFiles();

    for (int i = 0; i < files.length; i++)
    {
        File file = files[i];
        if (file.isDirectory())
            item.add(file.getName() + "/");
        else
            item.add(file.getName());
    }
    //Localizamos y llenamos la lista

    listFicheros.setVisibility(View.VISIBLE);
    ArrayAdapter<String> fileList = new
ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, item);
    listFicheros.setAdapter(fileList);

    listFicheros.setOnItemClickListener(new
```

```
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView,
View view, int i, long l) {
            tv1.setText(""+listFicheros.getItemAtPosition(i));
            onPause();

CargaFichero(""+listFicheros.getItemAtPosition(i));
            paso=0;
            listFicheros.setVisibility(View.INVISIBLE);
            onResume();
        }
    });
}
```

Como ya ha sido comentado, **Capturar**, en esta fase del TFG, no realizará ninguna función, ya que consistiría en la representación, en tiempo real, de los datos capturados por el sistema Arduino. No obstante, se ha preparado el sistema para que esté listo para la siguiente fase. De esta forma, al presionar el botón de captura, se inicializará el vector con valores nulos y se presentará un mensaje en pantalla avisando al usuario de que es necesario que realice todas las conexiones necesarias para la conexión Bluetooth. Este mensaje ha sido generado creando un layout específico para él en la carpeta "res" (mensaje_bt_ecg.xml).

```
public void onClickCapturar(View v){
     onPause();
     // limpiamos el array de datos y creamos un buffer con
muestras nulas para arrancar el gráfico
     datos.clear();
     paso=0;
     for(int k=0;k<1000;k++){datos.add((float)0);}
     LayoutInflater inflater = getLayoutInflater();
     View layout =
inflater.inflate(R.layout.mensaje_bt_ecg,null);
     Toast mensajeBT = new Toast(getApplicationContext());
     mensajeBT.setGravity(Gravity.CENTER | Gravity.TOP , 0,
100);
     mensajeBT.setDuration(Toast.LENGTH_LONG);
     mensajeBT.setView(layout);
     mensajeBT.show();
     boton_start.setVisibility(View.INVISIBLE);
     boton_stop.setVisibility(View.VISIBLE);
     onResume();
 }
```

Finalmente, **Volver** se realiza pulsando el botón creado con una imagen de una flecha, el cual finaliza la actividad actual mediante el método finish(), reanudando la actividad pausada antes de ser llamada (en este caso, la actividad Cardio).

3.2.2. – Pruebas y Verificación CardioGraficos

Para la actividad CardioGráficos el proceso de diseño y pruebas siguió el siguiente esquema:

- Prueba de la dependencia AndroidPlot: Realización del ejemplo suministrado para gráficos dinámicos.
- Prueba de carga de ficheros: Realización de aplicación sencilla que lea los ficheros capturados con Arduino y los escriba en pantalla.
- Prueba del interfaz: Construcción del layout de la actividad con representación de la señal de ejemplo y botones inactivos.
- Prueba de los botones Start y Stop: Comprobación de la implementación del código para pausar y reanudar la representación.
- Prueba carga y representación de archivos: Comprobación de la implementación del código de selección de archivos y modificación del código para representarlos
- Prueba/verificación de la actividad completa.

Durante esta etapa se aprendió el manejo de los controles básicos en Android (Buttom, TextView, ListView, etc) y la creación de layouts; el manejo de ficheros y la conversión de clases (de String a Float). También se han estudiado las posibilidades que ofrece la dependencia AndoidPlot y el uso de hilos o threads para la ejecución en paralelo de distintas acciones.

3.2.3. - Laboratorio

Es objetivo de esta parte de la aplicación se puede dividir en tres. Por un lado, la representación del ECG mediante la librería o dependencia AndroidPlot, de manera similar a la planteada en la actividad CardioGraficos. Por otro, el cálculo del ritmo cardiaco a partir de la señal del ECG. Finalmente, la capacidad de reproducir ficheros en formato mp3, basada en la librería MediaPlayer que trae el propio Android Studio. La intención del experimento es, una vez ya en la fase final, donde el ECG será obtenido en tiempo real, analizar como el carácter de la música puede modificar el ritmo cardiaco del usuario. En esta fase, no obstante, las señales analizadas provendrán de un fichero de texto, con ECG obtenido mediante el sistema Arduino comentado en el capítulo 2. El Layout de esta actividad se muestra en la Fig.27.

Fig. 27. Layout de la actividad CardioLab

Hilo Principal

Para el hilo principal se sigue la misma filosofía que en la actividad CardioGraficos. Se trabaja con un objeto XYPlot, y los datos se actualizan en un objeto de clase runnable SampleDynamicXYDatasource, el cual comunica al hilo principal cuando se modifican los datos mediante un modelo Observador-Observable. A diferencia de la actividad previamente mencionada, se incluye la carga de un fichero de texto con los datos de un ECG en el vector de datos directamente desde el inicio de la ejecución. Este vector emulará la señal de ECG que, en la última fase del TFG será obtenida directamente del Arduino y enviada por Bluetooth.

Se capturará también un nuevo control de tipo TextView, el cual será el encargado de representar en el ritmo cardiaco calculado.

```
pulso = (TextView) findViewById(R.id.textView6); //creamos el
TextView para actualizar el pulso
```

Hilo SampleDynamic

El método run() funciona de forma similar a de la actividad CardioGraficos. Se incrementa una posición (variable global "*paso*") la lectura del vector de datos, notificando al Observador cada 10 milisegundos, correspondiente al intervalo entre dos muestras del ECG obtenidas con el Arduino. En este caso el tamaño de representación ha sido elegido de 1000 muestras, correspondiente a 1000 segundos (10-20 latidos), ya que no se desea ver con tanto detalle el ECG y simplemente la representación sirve de apoyo al cálculo del ritmo cardiaco, objetivo final de esta actividad.

Este se calcula de manera sencilla gracias a que gran parte del acondicionamiento de la señal está ya realizado por el *hardware* del e-Health. Por tanto, simplemente se buscará el intervalo de tiempo entre dos complejos QRS mediante la superación de un umbral. Tras múltiples experimentos se ha llegado a la conclusión de situando este nivel en 2.5 V obtiene buenos resultados (la señal se encuentra centrada habitualmente sobre los 1.8 V y el pico del complejo QRS supera normalmente los 3V). Para evitar capturas erróneas, el procedimiento se espera cuarenta muestras antes de buscar el nuevo latido. Esto es equivalente a 0.4 segundos. Si los latidos se separaran menos de esta distancia no se podrían distinguir, no obstante, dicho ritmo equivaldría a 150 pulsaciones por minuto, ritmo que no se supera en condiciones normales. Las pulsaciones se calculan de manera simple a partir del intervalo transcurrido entre dos detecciones de umbral (Fig.28).

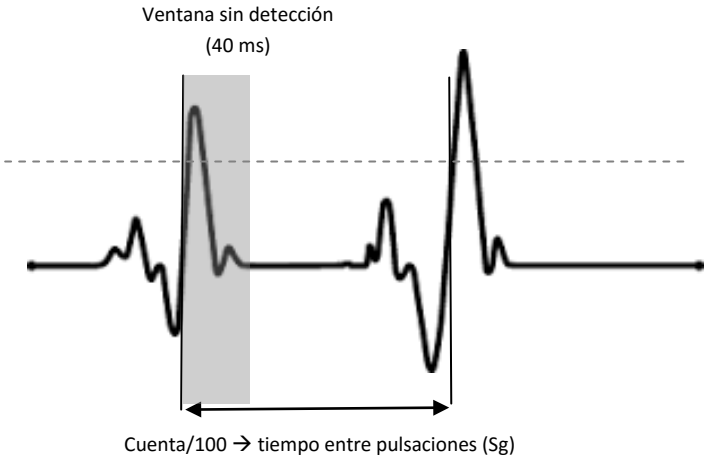


Fig. 28. Cálculo del ritmo cardiaco

Un punto clave de esta actividad es la actualización del valor del ritmo cardiaco en el control TextView. Como ya se ha visto, los controles pertenecen al hilo principal, mientras que el cálculo y actualización del ritmo cardiaco se realizan en el hilo SampleDynamic. Aunque el control TextView está definido como variable global, las actualizaciones que se realizaran no se representarían en el hilo principal hasta que el hilo SampleDynamic acabara (a efectos prácticos, nunca). Existen varias técnicas para comunicar datos entre los dos hilos. Una de las más utilizadas es el uso de manejadores o Handles. Otra es el uso de métodos post. Dado que el primero será usado para la comunicación Bluetooth, se ha optado por implementar el segundo en este caso y así trabajar ambos, ya que, en definitiva, el objetivo real de este TFG es el aprendizaje de la programación en Arduino y resulta interesante tocar el mayor número de aspectos del mismo.

El uso del método post es sencillo. Se aplica al objeto a modificar y contiene como argumento un objeto de tipo runnable cuyo método run consiste

únicamente en la actualización que se desea hacer en el objeto (en este caso, actualizar el texto del TextView con el valor actual del ritmo cardiaco).

Aunque este método es simple y útil para modificar un único control, en el caso de necesitar modificar o comunicar el estado de varios de estos se vuelve muy farragoso y el código más ilegible, por lo que en dichos casos se suele optar por la utilización de Handles.

```
try {
    keepRunning = true;
    while (keepRunning) {

        Thread.sleep(10); // Actualización cada 10 msg
        paso++;
        cuenta++;
        if (paso==datos.size()-SAMPLE_SIZE){paso=0;}
        // Estimación del ritmo cardiaco
        if (datos.get(paso)>2.5 && cuenta>40){ // si se la
supera 2.5 V y ha pasado más de medio segundo, estoy en el
siguiente latido
            PPM=60*100/cuenta;
            cuenta=0; // Reinicio la cuenta para el siguiente
latido

// Actualización del control TextView

           pulso.post(new Runnable() {
                @Override
                public void run() {pulso.setText("" + PPM);}
            });
        }         notifier.notifyObservers();
    }
} catch (InterruptedException e) {e.printStackTrace();}
}
```

Métodos OnClick

La actividad posee tres controles que pueden ser usados por el usuario, cada uno de ellos asociados a una respuesta:

- Relajación
- Activación
- Volver

Relajación y **Activación** permiten la reproducción de dos ficheros mp3 con músicas de ritmos adecuados con la idea de estudiar su efecto sobre el ritmo cardiaco. Esto se realiza mediante un objeto de la clase MediaPlayer ofrecido en el Android Studio. Simplemente invocando el método start() o stop() de este objeto se puede inicializar o detener la reproducción de cualquier fichero que se haya incluido entre los recursos de la aplicación (carpeta “res”, subcarpera “raw”). Se debe tener en cuenta que, para evitar la reproducción

simultanea de ambos ficheros de audio, antes de asignarle al objeto dicho fichero, se comprueba si su valor ya no es nulo, es decir, ya comenzó una reproducción, en cuyo caso se para antes de iniciar la nueva (en otro caso se escuchan las dos canciones a la vez).

```
public void PlayRelajacion(View v){
    if (mp !=null){
        mp.stop();}
    mp= MediaPlayer.create(this, R.raw.may_it_be);
    mp.start();
}

public void PlayActivacion(View v){
    if (mp !=null){
        mp.stop();}
    mp= MediaPlayer.create(this, R.raw.synchronicity);
    mp.start();
}
```

Finalmente, **Volver** se realiza nuevamente pulsando el botón creado con una imagen de una flecha, que invoca el método finish() de la actividad, reanudando la actividad previa, en este caso, la actividad Cardio). También se incluye el cierre del objeto MediaPlayer mediante el método reléase(), para detener la reproducción del audio y liberar recursos en caso necesario.

```
public void Volver(View v){
    if (mp !=null){
         mp.release();};
    finish();
}
```

3.2.4. – Pruebas y Verificación CardioLab

Para la actividad CardioLab el proceso de diseño y pruebas siguió el siguiente esquema:

- Prueba de la clase MediaPlayer: Reproducción de ficheros de audio. Reproducción alternada de dos ficheros de audio.
- Pruebas del método post(): Actualización de un TextView modicicado dentro de un Thread ejecutado en paralelo. Se utilizó el mismo esquema que en la actividad CardioGraficos, incluyendo un TextView que vaya representando la muestra actual.
- Prueba/verificación de la actividad completa

En esta fase se ha aprendido el uso de ficheros de audio a través de la clase MediaPlayer y la actualización de controles entre el hilo principal y un hilo ejecutado en paralelo mediante el método post.

3.3.- Electromiografía

La actividad Miograma (Fig. 29) presenta un layout similar a la actividad Cardio, con tres botones que permiten acceder tanto a la actividad MioGrafico de representación y captura (Gráfico), como a la actividad MioLab para el experimento (Laboratorio), así como retroceder a la actividad Main.

Fig. 29. Layout de la actividad MioGrama

3.3.1. - Representación

Se trata, básicamente, de la misma actividad Implementada en la actividad CardioGraficos, ya que comparte la misma funcionalidad, permitiendo la representación del EMG mediante la librería o dependencia AndroidPlot. Por lo tanto, comparte un layout muy similar (Fig.30).

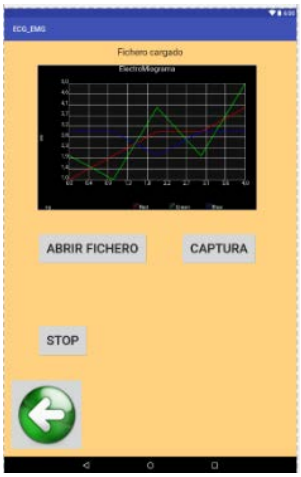


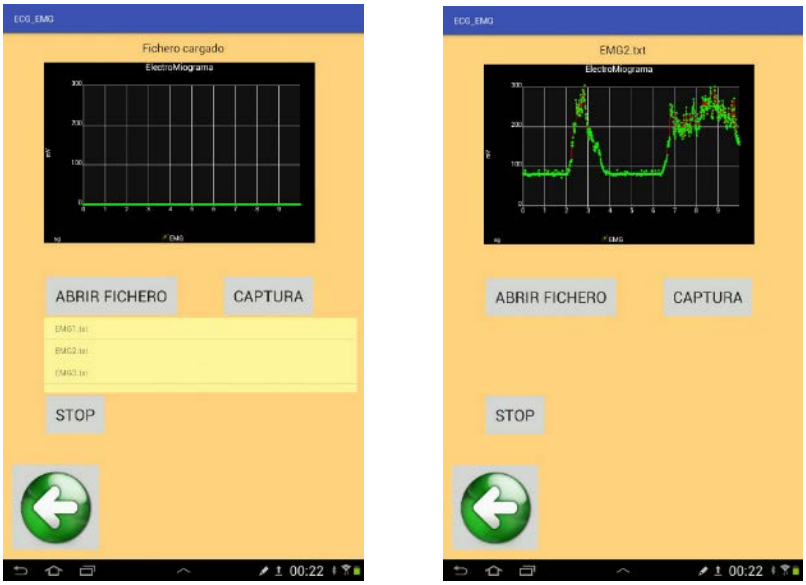
Fig. 30. Layout de la actividad MioGrafico

La actividad MioGrafico es idéntica a la actividad CardioGraficos, por tanto, mantiene la misma estructura señalada en la Fig. 25, con dos hilos funcionando en paralelo, el hilo principal y el hilo de SampleDynamic que va actualizando los datos a representar mediante un modelo Observador-Observable.

Como únicas modificaciones a señalar, hay que indicar que los ficheros de texto con capturas de EMG se encontrarán en este caso en otra carpeta, por lo que se ha modificado la variable String que almacena la dirección:

```
private String nomDirArch="/storage/sdcard0/IngSalud/EMG/";
```

Por otro lado, la duración de la representación gráfica se ha extendido de los 500 milisegundos hasta los 1500 milisegundos, ya que las señales de activación muscular suelen durar más que las señales del ECG. De este modo, su layout en búsqueda de ficheros y representación de los resultados sería la mostrada en la Fig. 31.



(a) (b)

Fig. 31. Carga de ficheros (a) y representación de un fichero cargado (b).

3.3.2. – Pruebas y Verificación

Dado que la actividad MioGráfico es idéntica en su base a la actividad CardioGrafico, el proceso de diseño ha consistido en modificar adecuadamente esta última actividad para incorporar las nuevas modificaciones. Por tanto, básicamente se ha probado el correcto funcionamiento de la actividad

completa. Se ha aprovechado que se disponía ya de seis actividades completas (Main, Cardio, Miograma, CardioGraficos y Miografico) para realizar la integración de todos en una única aplicación. Por tanto, las pruebas han sido:

- Prueba de actividad completa.
- Prueba de integración de todas las actividades.

En esta fase, fundamentalmente se ha aprendido a interrelacionar diferentes actividades en una única aplicación.

3.3.3. - Laboratorio

La actividad MioLabo, al contrario de lo ocurrido entre CardioGrafico y MioGrafico, es diferente a la actividad CardioLab en la parte del experimento. No obstante, también se basa en la representación del EMG mediante la librería o dependencia AndroidPlot por lo que comparte parte de su código. Su layout puede verse en la Fig.32. El objetivo de este experimento es representar gráficamente el nivel medio y la duración de los impulsos nerviosos captados en el EMG de manera dinámica.

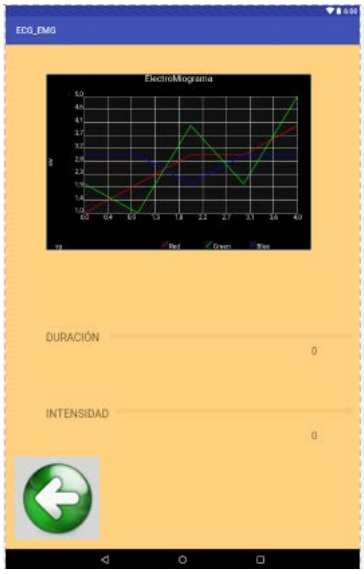


Fig. 32. Layout de la actividad MioLab

Hilo Principal

Para el hilo principal se sigue la misma filosofía que en la actividad CardioLabo. Nuevamente se usa un objeto XYPlot, siendo los datos se actualizados en un objeto SampleDynamicXYDatasource de tipo runnable, el cual comunica al hilo principal cuando se modifican los datos mediante un

modelo Observador-Observable. Al igual que en la actividad de laboratorio para los ECG, en esta fase se realiza la carga de un fichero de texto con los datos de un EMG para la emular el que se empleará en la última fase del TFG donde será obtenida directamente del Arduino y enviada por Bluetooth.

Para representar el nivel de la señal y su duración se usarán sendos controles TextView y otros dos de tipo progressBar. En este caso, al tratarse de cuatro controles en vez de uno solo como en la actividad CardioLab, se decidió realizar la captura de los controles en un método auxiliar como en CardioGraficos y MioGraficos.

```
public void capturarControles() {
    Duracion = (ProgressBar) findViewById(R.id.progressBar);
    Nivel = (ProgressBar) findViewById(R.id.progressBar2);
    Dur_text = (TextView) findViewById(R.id.textView8);
    Niv_text = (TextView) findViewById(R.id.textView9);
}
```

Hilo SampleDynamic

El método run() funciona de forma similar a de la actividad CardioLab. Se incrementa una posición (variable global "*paso*") la lectura del vector de datos cada 10 milisegundos correspondientes al intervalo entre dos muestras del EMG obtenidas con el Arduino. El tamaño de representación es de 1500 muestras correspondientes a un segundo y medio.

En este caso, a diferencia del ECG, el acondicionamiento de la señal mediante *hardware* por el e-Health no ha sido suficiente para procesar la señal, por lo que ha sido necesario incluir un nuevo filtrado paso bajo adicional (recordar aquí que la electrónica del e-Health incluye un filtrado paso bajo, uno paso alto y un rectificado, además de la amplificación).

La realización de este filtrado se realiza mediante el promediado de las diez últimas muestras de la señal del EMG. Para ello se genera un vector de diez muestras nulas (filtrado). Una vez dentro del bucle infinito del método run() se elimina el primer elemento del vector y se inserta al final el dato actual. De esta forma, el vector contendrá las ultimas diez muestras del EMG pudiéndose realizar el promediado.

Con esta información, para calcular el valor medio y la duración del impulso, se utiliza un umbral con histéresis, así, cuando se superan los 120 mV por primera vez se indica que se ha detectado un impulso y se mantiene en este estado hasta que la señal es inferior a 80 mV, en cuyo caso que considera acabado y se reinicializan todas las variables (Fig.33). Estos umbrales han

mostrado se útiles en la mayoría de los experimentos realizados. No obstante, se recuerda que la extensión e-Health posee, para la medida del EMG, de un amplificador de ganancia variable, con lo que se puede ajustar en caso necesario para que los niveles de la señal se ajusten a los umbrales.

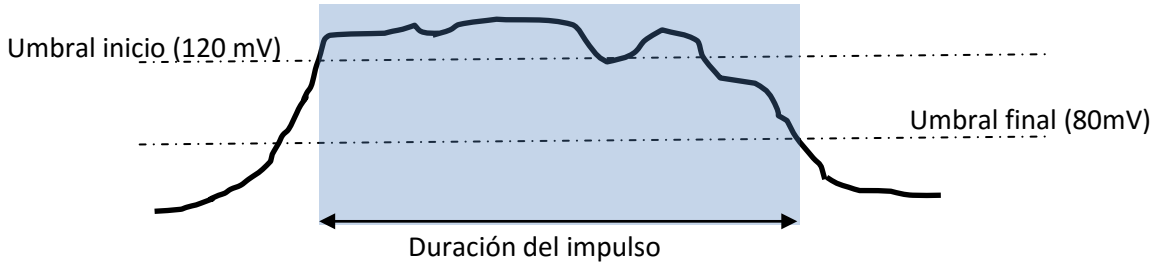


Fig. 33. Uso de umbrales con histéresis para el cálculo de la duración del impulso.

Mientras se está en el impulso, se actualizarán los controles de la actividad (ProgressBar y TextView respectivos), de tal forma que irán variando dinámicamente mientras se mantenga la contracción y, al acabar ésta, los controles quedarán fijos a sus valores máximos hasta la llegada de un nuevo impulso. Para la realización de estas actualizaciones se utiliza el método post ya explicado para la actividad CardioLab. Como se puede observar en el código adjunto éste comienza a ser mucho más extenso, repetitivo y poco elegante al actualizar cuatro controles.

```
try {
    keepRunning = true;

// inicializar el vector para el filtrado

for (int k=0;k<10;k++){filtrado.add((float)0);}


 while (keepRunning) {

     Thread.sleep(10); // Actualización cada 10 msg
     paso++;
     if (paso==datos.size()-SAMPLE_SIZE){paso=0;}


     //Filtrado Paso bajo

     dato=datos.get(paso);
     filtrado.removeFirst();
     filtrado.addLast(dato);
     for (int k=0;k<10;k++){dato= dato+ filtrado.get(k);}
     dato=dato/10;


     if (impulso && dato>120) { // si ya se esta en un impulso
 y no se baja de 100 mV, se procesa el dato

         marca++;
         sum_nivel=sum_nivel+dato;
         nivel_med=sum_nivel/marca;
```

```
            Duracion.post(new Runnable() {
                @Override
                public void run() {
                    Duracion.setProgress(marca);
                }
            });
            Dur_text.post(new Runnable() {
                @Override
                public void run() {
                    Dur_text.setText("" + marca*10 + " msg");
                }
            });

            Nivel.post(new Runnable() {
                @Override
                public void run() {
                    Nivel.setProgress((int)nivel_med);
                }
            });
            Niv_text.post(new Runnable() {
                @Override
                public void run() {
                    Niv_text.setText("" + (int)nivel_med + " mV");
                }
            });
        }

        if (datos.get(paso)> 120 && !impulso){  // si se la
    superan 100 mV y no era un impulso comienza dicho impulso
            impulso=true;}

        if(impulso && datos.get(paso)<=80){ // si ya se esta en un
    impulso y se baja de 20 mV, ha acabado
            marca=0;
            sum_nivel=0;
            impulso=false;
        } // Reinicio la cuenta para el siguiente impulso
```

MÉTODOS OnClick

El único método relacionado con la pulsación de un control es el que permite finalizar la actividad (**Volver**) y regresar a la anterior, Miograma.

3.3.4. – Pruebas y Verificación

Aunque su filosofía es muy diferente a la de la actividad CardioLabo, no incluye muchos conceptos de programación nuevos por lo que no ha sido necesario verificar grandes aspectos. En la práctica se modificó CardioLabo para trabajar con señales de EMG y se utilizó el control TextView utilizado para la representación del ritmo cardiaco como visor para verificar el cálculo correcto de los resultados intermedios del proceso de filtrado y de la intensidad media y

la duración de los impulsos nerviosos. Finalmente se incluyeron los controles de ProgressBar y TextView finales.

4

SISTEMA DE COMUNICACIONES BLUETOOTH

Este cuarto Capítulo contiene la descripción del sistema necesario para la comunicación inalámbrica entre el sistema Arduino y la aplicación móvil. Para ello se realizará un sistema sencillo en el que una aplicación simple emulará las funcionalidades necesarias. Éstas consisten en el envío de la información por parte del sistema Arduino (que serán los datos del ECG o el EMG) y la recepción de estos en una aplicación de Android. Posteriormente, una vez verificado su correcto funcionamiento, se incorporarán estas modificaciones al sistema completo. Dado que la creación de la comunicación Bluetooth conlleva múltiples aspectos interesantes por sí mismos y ha consumido gran parte del esfuerzo dedicado al TFG, se desarrollará en este capítulo tanto las modificaciones incluidas en la aplicación final de forma separada para separar los distintos conceptos, como, previamente y con detalle la aplicación básica y los distintos pasos que se han ido realizando. El código completo se encuentra en el CD adjunto.

4.1.- Estructura Básica de una comunicación Bluetooth

La comunicación inalámbrica entre el sistema Arduino y la aplicación móvil en Android debe enviar los datos que se están capturando mediante la extensión e-Health a la aplicación Android. Antes de realizar esta incorporación al sistema presentado en los capítulos dos y tres se realizó una aplicación sencilla consistente en enviar una serie de números generados con Arduino para ser representados visualizados de manera simple en una aplicación Android.

No obstante, es necesario tener en cuenta que, antes de plantearse el realizar ningún tipo de programación, es necesario realizar una serie de pasos para configurar la extensión de comunicaciones de Arduino y el módulo Bluetooth PRO y comprobar el correcto funcionamiento de dicho módulo.

Configuración básica y prueba de conectividad

Para ello, en primer lugar, se realizará una conexión con el modulo Bluetooth PRO haciendo uso de un emulador de puerto serie por USB en el PC (en el caso del presente TFG, se ha utilizado el Hterm 0.8.1 [13]). Por el otro lado, es necesario recordar que para utilizar el módulo Bluetooth PRO existen dos modos de funcionamiento (XBEE y USB) seleccionables mediante el uso de un puente (Fig.34), siendo el segundo de ellos el que permite la comunicación directa del PC con el módulo de comunicaciones.

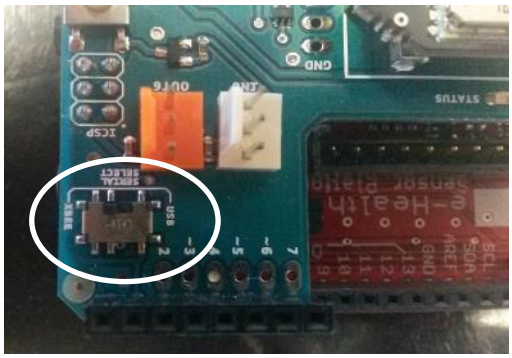


Fig.34. Interruptor selector de modo Shield de comunicaciones.

De esta forma, se podrán mandar comandos al módulo y se podrá recibir aquella información recibida por el mismo, aunque no se podrá comunicar con el Arduino. Aunque en [6] se indica que en este modo se debería extraer el microcontrolador del Arduino UNO, en otros tutoriales consultados se comenta que es suficiente con instalar en el mismo el denominado programa “bare minimun”, consistente básicamente en un sketch que no realiza nada:

```
void setup() {
}
void loop() {
}
```

Por lo tanto, los pasos a seguir son: primero cargar el bare mininimum en Arduino sin la extensión de comunicaciones conectada y a continuación, conectar ésta en modo USB. A continuación, se puede ejecutar el emulador de puerto serie Hterm. En interfaz de la aplicación (Fig.35), se debe elegir el puerto de comunicaciones que está usando el Arduino (recordar que se habrá seleccionado en el momento de instalarlo, en el caso de este TFG, el COMM3). También se ha de seleccionar la velocidad en baudios. En necesario reseñar aquí que, aunque en los tutoriales de [14] se indica que la velocidad por defecto es de 38400 baudios, esto no es cierto, siendo necesario configurarla a 115200 baudios si se quieren usar dichos valores por defectos. Para finalizar, seleccionar 8 bits con 1 de stop sin paridad y envío del carácter de retorno de carro (CR) al pulsar “enter” para enviar.

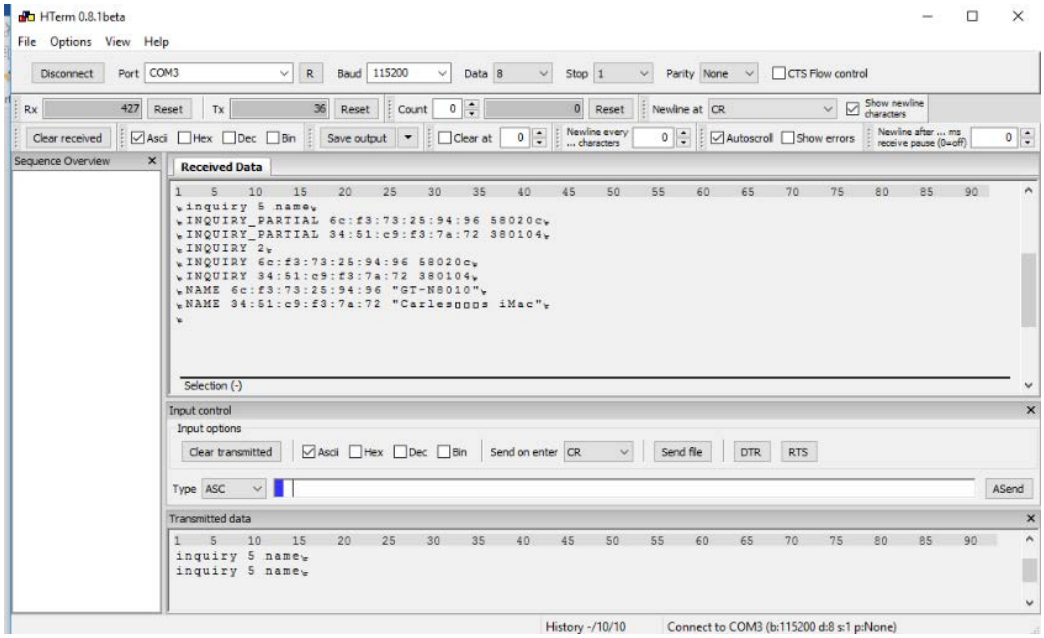


Fig.35. Interfaz de la aplicación Hterm 0.8.1.

A partir de aquí se pueden enviar comandos al módulo Bluetooth para comprobar su correcto funcionamiento. Por ejemplo, enviando “AT” veremos cómo éste nos responde con un “OK”. Lo siguiente sería comprobar que el módulo es capaz de localizar la Tablet para intentar conectarse con ella. Esto se puede hacer enviando en comando “INQUIRY {tiempo de espera}”, que nos devuelve los dispositivos visibles con conexión Bluetooth durante el {tiempo de espera} (hay que recordar hacer visible el dispositivo objetivo). Si al comando añadimos “name”, nos devolverá, además de las direcciones MAC de los dispositivos, su tipo y la potencia recibida, el nombre del dispositivo dado por

su propietario, lo que nos hará más fácil localizar nuestro dispositivo [14]. En nuestro caso, la tablet, con el nombre por defecto “GT-N8010” fue fácilmente localizada.

Una vez comprobado que el módulo funciona correctamente y es capaz de localizar la tablet se podría realizar la vinculación de ambos vía programación. No obstante, dado que el dispositivo tablet ya tiene incorporado todo el protocolo para realizar la vinculación entre los dispositivos se consideró mucho más sencillo realizar ésta haciendo uso de dicha facilidad. Para ello, lo único necesario era darle al módulo Bluetooth de Arduino un PIN, ya que, por defecto no tiene ninguno. Esto se realiza mediante el envío del comando SET CT AUTH * 1234 (donde 1234 sería el PIN). Una vez hecho esto, desde el dispositivo se puede localizar el módulo Bluetooth (por defecto su nombre es WT12-A y siempre es visible) y realizar la vinculación introduciendo el PIN creado.

Si se desease, se pueden cambiar múltiples parámetros configuración del módulo Bluetooth mediante los comandos SET. Por ejemplo, el nombre público del dispositivo (SET BT NAME nombredisp) o la velocidad de comunicación (CONTROL BAUD 115200,8n1, para 115200 baudios, 8 bits con 1 de stop sin paridad). No obstante, para esta aplicación, se dejaron los valores por defecto.

Vinculados ya ambos dispositivos, se instaló en el dispositivo Tablet la aplicación BT Simple Terminal (Fig.36), el cual hace una función similar a Hterm, pero para dispositivos Bluetooth vinculados al dispositivo. La instalación para dispositivos Android es sencilla desde su dirección descarga en Google Play [15].

Una vez seleccionando el dispositivo vinculado deseado (WT12-A), disponemos de dos emuladores de puerto serie en los extremos de la comunicación, pudiendo enviar datos por cualquiera de los dispositivos siendo recibidos correctamente por el otro.

Fig.36. Interfaz de la aplicación BT Simple Terminal.

Una vez comprobado que las comunicaciones Bluetooth pueden realizarse sin problemas entre el dispositivo Tablet y el sistema Arduino, se implementaron sendos programas en ambas partes para que se estableciese la conexión y la comunicación de manera autónoma.

Sección Arduino

Por parte del sistema Arduino, el desarrollo consistió en un programa muy simple dedicado a enviar una serie de números consecutivos a intervalos de 100 milisegundos entre cifra y cifra. Para hacerlo lo más similar al sistema final, estos datos se generan del tipo float, como los datos obtenidos para el ECG y EMG, y se enviarán con la misma resolución de tres decimales. No se utilizan 10 milisegundos para poder apreciar el cambio de las cifras enviadas en el dispositivo receptor. Como el módulo de comunicaciones se deja con los parámetros por defecto, la velocidad de envío será de 115200 baudios.

```
int cont=0;

void setup() {
Serial.begin(115200);
}

void loop() {
   float dato=(float)cont/1000;
   Serial.print(dato,3);
   Serial.println("");
   cont++;
   delay(100);
}
```

Para que el microcontrolador los envíe por el módulo de comunicaciones, simplemente se ha de programar un envío por el puerto serie, teniendo el módulo de comunicaciones configurado en modo XBEE. En la tablet se siguió usando el BT Simple Terminal para comprobar el correcto envío de los datos. De esta manera, los pasos a seguir fueron:

- Cargar el sketch en Arduino (sin el módulo de comunicaciones conectado)
- Conectar el módulo de comunicaciones en modo XBEE y alimentar el Arduino para que comience la ejecución del sketch.
- Ejecutar BT Simple Terminal en la Tablet y seleccionar el dispositivo Bluetooth de Arduino para realizar la comunicación.

En el dispositivo tablet, una vez seleccionado en la aplicación BT Simple Terminal el dispositivo Arduino entre aquellos vinculados, se observa cómo se van recibiendo los datos generados en el módulo Arduino de manera correcta,

verificándose el correcto funcionamiento del módulo Bluetooth y el envío de datos.

Sección Android

El primer paso necesario para incluir cualquier uso de las comunicaciones Bluetooth en una aplicación Android es incluir en el Manifest del proyecto los permisos necesarios

```
<uses-permission android:name="android.permission.BLUETOOTH"/>

<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

El primero de ellos (BLUETOOTH), permite únicamente realizar conexiones Bluetooth y transferir datos. El segundo (BLUETOOTH_ADMIN), permite, además de realizar conexiones Bluetooth y transferencias de datos, manipular las opciones del sistema en lo referente a Bluetooth, buscar otros dispositivos y realizar vínculos. Aunque en sentido estricto solo es necesario el primero de ellos para la aplicación deseada, se han incluido los dos para futuras extensiones.

Como layout para la aplicación se optó por un interfaz muy simple. Incluye un control tipo TextView donde se representarán los datos recibidos y un ListView, que nos mostrará la lista de dispositivos vinculados para seleccionar con cual haremos la conexión (Fig. 37).

Fig.37. Layout del programa de prueba de conexión Bluetooth

En lo que respecta a la programación propiamente dicha, es necesario tener en cuenta que se necesitan dos hilos. Uno que llevará el programa principal y los controles y otro hilo que se encargará de ir consultando la

conexión Bluetooth y tomando los datos correspondientes. Pero antes de lanzar este segundo hilo se va a buscar un dispositivo vinculado para realizar la conexión. Lo primero es crear un adaptador para acceder al controlador Bluetooth del dispositivo. Si este objeto tuviera valor nulo, significaría que el dispositivo no tiene control Bluetooth, por lo que se finaliza la actividad ya que no tiene sentido continuarla.

```
miBTadaptador = BluetoothAdapter.getDefaultAdapter();

if (miBTadaptador == null) {
    Toast.makeText(this, "No hay Bluetooth disponible en este
dispositivo", Toast.LENGTH_LONG).show();
    finish();
    return;
}
```

Una vez obtenido el adaptador, se puede obtener un conjunto (set) con los dispositivos vinculados. Los nombres de estos dispositivos son introducidos en un array para que puedan ser pasados al ListView y ser mostrados al usuario.

```
pairedDevices = miBTadaptador.getBondedDevices();
ArrayList list = new ArrayList();
for(BluetoothDevice bt : pairedDevices)
    list.add(bt.getName());
final ArrayAdapter adapter = new
ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
lv.setAdapter(adapter);
```

Cuando el usuario selecciona un dispositivo de la lista se obtiene el objeto correspondiente al dispositivo con dicho nombre y con él se crea un socket con el método createRfcommSocketToServiceRecord (el identificador UUID utilizado es el estándar por defecto). Obtenido el socket se crea un objeto de tipo BluetoothService que se encargará de generar el hilo paralelo y gestionar la comunicación.

```
lv.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
   @Override
   public void onItemClick(AdapterView<?> adapterView, View
view, int i, long l) {
       Object nombDispo = lv.getAdapter().getItem(i);
       for (BluetoothDevice bt : pairedDevices) {
           if (nombDispo.equals(bt.getName())) {
               dispositivo = bt;
               break;
           }
       }
        try {
            UUID uuid = UUID.fromString("00001101-0000-1000-
8000-00805f9b34fb");
            miSocket =
```

```
dispositivo.createRfcommSocketToServiceRecord(uuid);
        } catch (IOException e) {
        }
        try {
            miSocket.connect();
        }
        catch(IOException e) {
        }
        if (miSocket != null) {
            lv.setVisibility(View.INVISIBLE);
        }
        BluetoothService servicio= new
BluetoothService(manejador, miSocket);
        servicio.realizarConexion();
    }
```

La clase BluetoothService se ha creado de manera independiente y tiene tres variables: el socket, el manejador (handler) que permitirá el paso de datos entre los hilos y un objeto de tipo HiloConexion que será de tipo runnable y generará el hilo paralelo. Así mismo contiene dos únicos métodos además del constructor: crearConexion() y finalizarServicio().

El primero de ellos crea el objeto HiloConexion y arranca el segundo hilo. En el constructor de los objetos de esta clase se genera un objeto del tipo InputStream que se encargará de adquirir todos los datos recibidos por Bluetooth, lo cual se realizará en el método run() de esta clase. En él, cada 100 milisegundos (ritmo al que se envían los datos desde Arduino) se toman los datos obtenidos del socket con un BufferedReader, se obtiene una línea (significaría que hay un dato disponible) y se envía con el manejador al hilo principal. Gracias al BufferedReader no es necesario que ambos sistemas estén sincronizados ya que siempre se envían al hilo principal los datos cuando estén completos. Lo que si es necesario es que ambos sistemas tarden lo mismo en enviar/recibir datos, ya que, si el sistema Arduino envía datos a mayor velocidad se irán acumulando los datos en el buffer, ya que cada intervalo en el sistema Android se obtiene un único dato.

```
public void run()
{
    byte[] buffer = new byte[1024];
    int bytes;
    InputStreamReader lectorDatos = new
InputStreamReader(inputStream);
    BufferedReader br = new BufferedReader(lectorDatos);

    // Mientras se mantenga la conexion el hilo se mantiene en
espera ocupada leyendo del flujo de entrada
    while(true)
    {
        try {
            // Leemos del flujo de entrada del socket
            //bytes = inputStream.read(buffer);
              String texto = br.readLine();
```

```
            // Enviamos la informacion a la actividad a traves del
            handler.
            // El metodo handleMessage sera el encargado de recibir el
            mensaje y mostrar los datos recibidos en el TextView
                        handler.obtainMessage(1,texto).sendToTarget();
                        sleep(100);
                    }
                    catch(IOException e) {}
                    catch (InterruptedException e) {e.printStackTrace();}
                }
            }
```

Para terminar de comprender como funciona la aplicación es necesario hablar del manejador o handle, Este es creado en el hilo principal y enviado al hilo secundario de la comunicación con el constructor del objeto HiloConexion. De esta forma, mediante el método obtainmessage().sendToTarget() se envía el dato al hilo principal y se ejecuta con el dato obtenido lo que se desee tal y como esté definido en el método handleMessage() del manejador, en este caso, la aparición del dato obtenido en el TextView del hilo principal.

```
        private final Handler manejador = new Handler() {

            @Override
            public void handleMessage(Message msg)
            {
                byte[] buffer  = null;
                String mensaje     = null;

                 // Mensaje de lectura: se mostrara en el TextView

                        mensaje = (String)msg.obj;
                        tvMensaje.setText(mensaje);
            }
        };
```

Ejecutando esta aplicación, en primer lugar se solicitará que elijamos el dispositivo vinculado con el que queremos hacer la conexión. Una vez realizada la conexión, irá representando los datos recogidos en el control TextView. El sistema se ha probado con el sketch Arduino antes mencionado observándose la correcta aparición de los números esperados. Indicar solamente que, en el caso de dejar produciéndose números de manera indefinida llega un momento en el que se produce aparentemente un desbordamiento del buffer y el programa se bloquea. Esto nos lleva a suponer que el programa de Arduino funciona ligeramente más rápido que el de Android. Esto es lógico, ya que el número y complejidad de operaciones realizados en cada intervalo por Android son claramente mayores que las realizadas por Arduino, por lo que, aunque ambos poseen el mismo retardo *software*, el segundo tarde un poco más en realizar el resto de instrucciones de cada bucle.

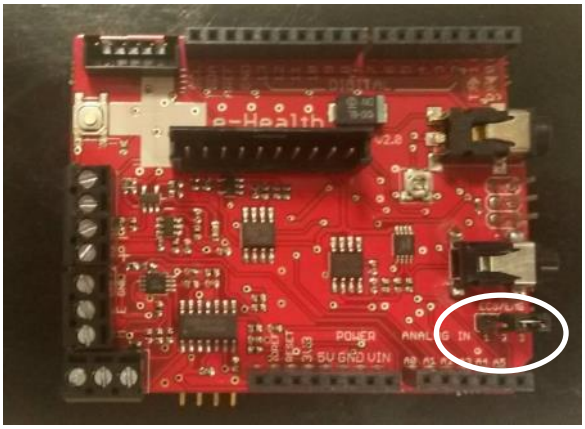
4.2.- Modificación y extensión de la aplicación Arduino

Como se acaba de ver, para incluir la comunicación Bluetooth no es necesario incluir ninguna modificación especial, ya que, una vez conectado el módulo de comunicaciones en el modo XBEE, la salida del puerto serie se trata directamente como la entrada de dicho módulo.

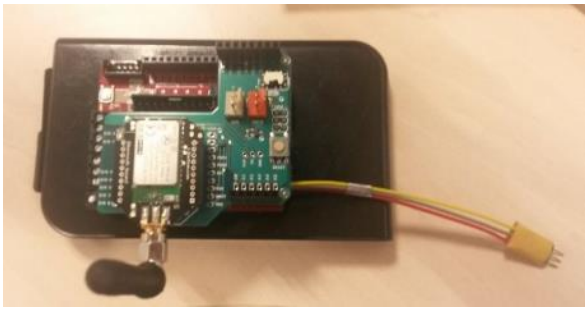
No obstante, a la hora de conectar físicamente el módulo de la extensión de comunicaciones sobre el e-Health, los pines de 5V y GND, así como el puente selector de la medida ECG o EMG de la extensión e-Health quedan ocultos por dicho módulo. Se hace necesario por tanto realizar una conexión externa para el puente y generar las señales de nivel alto y bajo en pines accesibles por el usuario. El pin con el nivel bajo de señal es accesible directamente desde el pin digital 14. El nivel alto se obtiene colocando el pin 13 como salida digital y fijándolo a nivel alto por *software*.

```
#include <eHealth.h>
int cont=0;
int pinSeleccion=2;
int pinAlto=13;

void setup() {
Serial.begin(115200);
pinMode(pinSeleccion,INPUT); //Pin 2 HIGH, medida del ECG
                             //Pin 2 LOW, medida EMG
pinMode(pinAlto,OUTPUT);
digitalWrite(pinAlto,HIGH);
delay(3000);
}

void loop() {
 if (digitalRead(pinSeleccion)==LOW){
   delay(10);
   float ECG =eHealth.getECG();
   Serial.print(ECG, 3);
   Serial.println("");
    }
  else {
      delay(10);
      float EMG =eHealth.getEMG();
      Serial.print(EMG, 3);
      Serial.println("");
     }
}
```

Para la extensión del puente selector se han utilizado un trío de latiguillos de tal forma que pueda ser accesible desde el exterior (Fig. 38)



(a)



(b)

Fig.38. Puente de selección ECG/EMG (a) y prolongación del mismo para acceso con el shield e-Health (b).

Para comprobar el correcto envío de los datos sin necesidad de implementar las modificaciones en la parte Android se ha utilizado la aplicación BT Simple Terminal. Sin embargo, se observó que, en esta ocasión no se producía la conexión, indicándose que el dispositivo WT12-A se encontraba no disponible. Tras varias pruebas se llegó a la conclusión de que, si el sistema Arduino se encuentra generando los datos y colocándolos en el módulo Bluetooth cada 10 milisegundos, dicho módulo es incapaz de realizar las tareas de establecimiento de la comunicación. Es por ello que se necesita que, mientras el modulo esté estableciendo la comunicación, no se estén generando datos. Para ello se ha introducido un retardo de dos segundos en el bloque setup() del sketch. De esta manera, los pasos para establecer la comunicación serían:

- Una vez cargado el sketch en Anduino, solicitar la conexión Bluetooth en la aplicación Android.
- Pulsar inmediatamente el botón de *reset* del Arduino, con lo que se reinicializa su sketch y entra en los dos segundos de pausa del setup().

- Se observa como en este caso sí se establece la comunicación y pasados un segundo más o menos, los datos comienzan a aparecer en el TextView.

4.3.- Pruebas y verificación

Para el sistema Arduino final de captura de las señales biológicas el proceso de diseño y pruebas siguió el siguiente esquema:

- Prueba de funcionamiento sin envío: conexión y ejecución del sketch del sistema en conexión tipo USB. De esta forma se ve en el puerto serie que los datos son generados correctamente y que la extensión del puente selector ECG y EMG y los nuevos pines de selección del bucle de medida funciona correctamente
- Prueba de envío: Prueba del envío y recepción de los datos con BT Simple Term en el terminal Android.
- Prueba completa: Envío más cambio de señal a medir.

Como se ha comentado, en esta fase se ha detectado los problemas de conexión al generarse los datos cada 10 milisegundos en el sistema Arduino. La solución adoptada mediante la inclusión de un retraso en el setup() del sketch y el *reset* del mismo a la hora de hacer la conexión ha sido igualmente desarrollada en el epígrafe anterior. No obstante, en las pruebas realizadas se ha observado que, a partir de intervalos de 50 milisegundos en el envío de los datos, la conexión se puede establecer sin problemas (por eso no se apreció este problema en las pruebas previas al usarse 100 milisegundos). Este valor puede ser aceptable para el EMG si lo único que se quiere obtener es su duración e intensidad media, pero en el caso del ECG, implica un muestreo insuficiente. Es por ello que se opta por dejar los 10 milisegundos de intervalo y mantener la solución del reinicio del sketch. Existen soluciones más complejas y elegantes como se comenta en el capítulo 5, pero en esta fase del TFG se ha optado por esta solución más sencilla.

4.4.- Modificación y extensión del sistema Android

Por su lado, el sistema Android sí que necesita múltiples modificaciones para su correcto funcionamiento. Por un lado, necesita incluir la programación necesaria para realizar la comunicación Bluetooth con el sistema Arduino. Pero además se necesita modificar la programación de las diferentes actividades para que hagan uso de estos datos en vez de los archivos de texto comentados

en el capítulo 3. Finalmente, dado que también se va a incluir la funcionalidad de grabar en un fichero de texto las señales adquiridas, será necesario incluir los métodos que realicen dicha acción.

Inclusión de la comunicación Bluetooth:

La comunicación mediante Bluetooth será incluida siguiendo la misma filosofía presentada en la aplicación de prueba, es decir, la captura de los datos se hará en una clase auxiliar que incorpora una variable tipo runnable, que creará un hilo paralelo donde se estarán esperando los datos. El paso de estos al hilo principal se hará mediante un manejador, que se creará en cada una de las actividades que van a realizar capturas en tiempo real de los datos (CardioGraficos, CardioLab, MioGrafico yMioLab). En esta ocasión, lo que realizará dicho manejador es la adición del dato capturado en el vector de datos que está siendo representado en el gráfico. Dado que el número es recibido como una cadena de caracteres, es necesario realizar su transformación al tipo float.

```
private final Handler manejador = new Handler() {

    @Override
    public void handleMessage(Message msg)
    {
        byte[] buffer  = null;
        String mensaje     = null;

        // Mensaje de lectura: se mostrara en el TextView

        mensaje = (String)msg.obj;
        datos.add(Float.parseFloat(mensaje));
    }
};
```

La clase auxiliar BluetoothService queda exactamente igual que en el ejemplo simple visto en epígrafes anteriores, ya que se encarga de realizar el hilo paralelo para la recepción de los datos de igual forma que antes. El único cambio necesario es, obviamente, disminuir al intervalo a los 10 milisegundos, ya que esta es la tasa de envío de las señales ECG y EMG del sistema Arduino.

Modificación de las actividades:

Aunque van a compartir gran parte de las modificaciones, hay que distinguir fundamentalmente entre aquellas realizadas en las actividades tipo “Gráfico” y aquellas tipo “Laboratorio”.

Para las primeras, el layout se modificará incluyendo un control tipo botón y otro tipo EditText que permitirán la grabación de los resultados finales (Fig. 39).

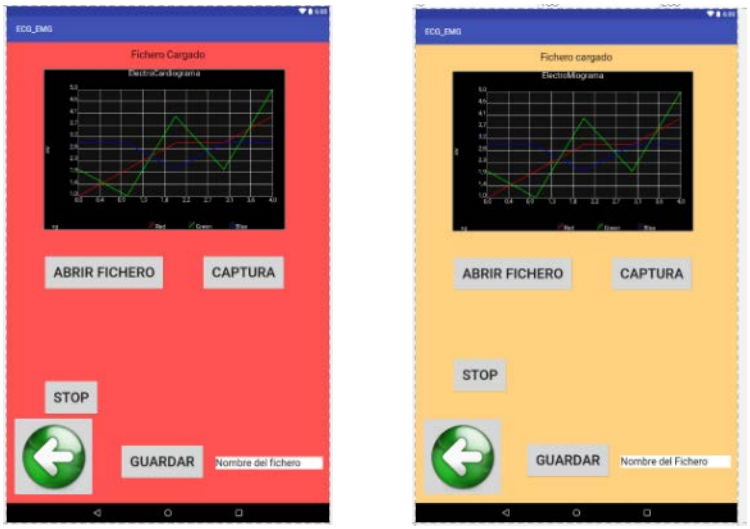


Fig.39. Layout definitivo de las actividades CardioGraficos y MioGrafico.

El código de la actividad en lo referente al hilo paralelo de representación de datos a partir de ficheros queda exactamente igual. La acción onClick del botón Captura sí debe ser, lógicamente, modificado. En esta ocasión se realizan los siguientes pasos:

- Se detiene la representación de datos.
- Se limpia el vector de datos (se eliminan todos los datos).
- Se llena el vector de datos con 600 muestras nulas para el ECG (o 1600 para el EMG). Esto permite tener un buffer de datos mientras se reciben las primeras capturas ya que el gráfico necesita 500 muestras disponibles (1500 en el EMG) para representar desde el primer instante.
- Se actualiza la visibilidad de los controles: Se ocultan los botones de Start-Stop y se hace visible el ListView. Para ahorrar controles, se va a usar el mismo control que se usaba para listar los ficheros representables con el objetivo de representar los diversos dispositivos Bluetooth vinculados.
- Se crea la conexión Bluetooth tal y como se ha visto en la prueba básica del epígrafe 4.1 en su apartado de Android, con lo que comienza el hilo paralelo de captura de datos.
- Se hacen visibles los controles relacionados con el guardado de los datos y se oculta el ListView.
- Se reactiva el hilo de representación de datos.

```
public void onClickCapturar(View v){
    onPause();
    // limpiamos el array de datos y creamos un buffer con
```

```
        muestras nulas para arrancar el gráfico
            datos.clear();
            paso=0;
            for(int k=0;k<600;k++){datos.add((float)0);} // k<1100 para
        el caso del EMG


            LayoutInflater inflater = getLayoutInflater();
            View layout =
        inflater.inflate(R.layout.mensaje_bt_ecg,null);
            Toast mensajeBT = new Toast(getApplicationContext());
            mensajeBT.setGravity(Gravity.CENTER | Gravity.TOP , 0, 100);
            mensajeBT.setDuration(Toast.LENGTH_LONG);
            mensajeBT.setView(layout);
            mensajeBT.show();
            boton_start.setVisibility(View.INVISIBLE);
            boton_stop.setVisibility(View.INVISIBLE);
            listFicheros.setVisibility(View.VISIBLE);

            // Obtención de la lista de dispositivos vinculados

            miBTadaptador = BluetoothAdapter.getDefaultAdapter();
            pairedDevices = miBTadaptador.getBondedDevices();
            ArrayList list = new ArrayList();

            for(BluetoothDevice bt : pairedDevices)
                list.add(bt.getName());
            final ArrayAdapter adapter = new
        ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
            listFicheros.setAdapter(adapter);
            listFicheros.setOnItemClickListener(new
        AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> adapterView, View
        view, int i, long l) {
                    Object nombDispo =
        listFicheros.getAdapter().getItem(i);
                    for (BluetoothDevice bt : pairedDevices) {
                        if (nombDispo.equals(bt.getName())) {
                            dispositivo = bt;
                            break;
                        }
                    }
                    try {
                        UUID uuid = UUID.fromString("00001101-0000-1000-
        8000-00805f9b34fb");
                        miSocket =
        dispositivo.createRfcommSocketToServiceRecord(uuid);
                    } catch (IOException e) {
                    }
                    try {
                        miSocket.connect();
                    }
                    catch(IOException e) {
                    }
                    if (miSocket != null) {
                        listFicheros.setVisibility(View.INVISIBLE);

                    }

                    servicio= new BluetoothService(manejador, miSocket);
```

```
                servicio.realizarConexion();

            }

        });
        boton_guardar.setVisibility(View.VISIBLE);
        nomarchivo_text.setVisibility(View.VISIBLE);
        onResume();
    }
```

De esta forma, una vez pulsado el botón de Capturar y seleccionado el dispositivo Bluetooth (WT12-A), los datos comenzarán a representarse en el gráfico, comenzando con una línea plana en el cero. La captura terminará cuando se pulse el botón de guardar o bien cuando se pulse el botón de Volver. En este caso, dado que la conexión Bluetooth podría estar todavía abierta, por lo que se comprueba y, en caso de ser necesario, se cierra.

```
public void Volver(View v) {
    if (servicio !=null){
        servicio.finalizarServicio();}
    finish();
}
```

En el caso de las actividades de tipo Laboratorio, el código y la filosofía seguidas son muy parecidas. Los layout se modifican incluyendo los controles dedicados al guardado de los datos (Fig.40). En este caso también se incluye un ListView para el listado de los dispositivos Bluetooth, ya que no se disponía de ninguno (no se listaban ficheros). De inicio, en el caso de CardioLab se ocultan los controles relacionados con el Ritmo Cardiaco y los botones de Relajación y Activación, mientras que en MioLab se ocultan los controles ProgressBar y TextView de Duración e Intensidad. Esto se realiza de esta manera ya que es necesario el espacio para la lista de dispositivos vinculados.

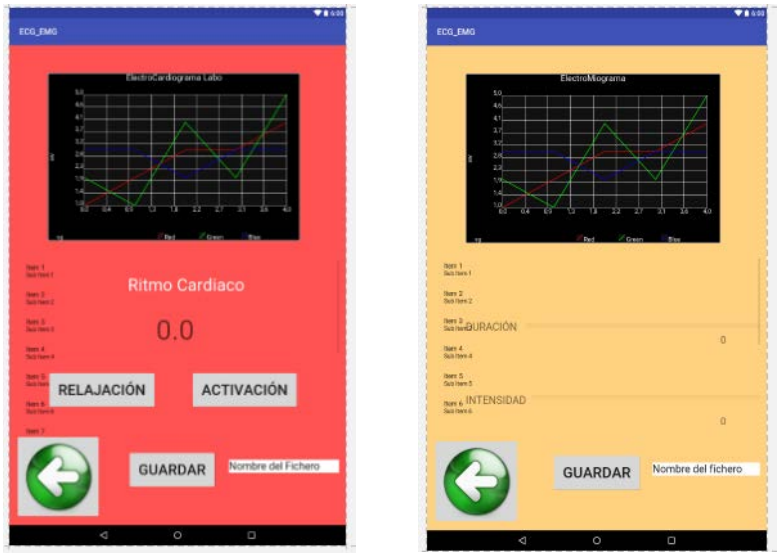


Fig.40. Layout definitivo de las actividades CardioLabo y MioLabo.

La principal diferencia con las actividades de tipo “Gráfico” es que, en este caso, al realizarse siempre la conexión Bluetooth desde el inicio, todo el código relacionado con ello se encuentra incluido directamente en la clase principal en vez de estar incluido en el método relacionado con la pulsación de un botón (Capturar). Una vez realizada la conexión, se oculta el ListView y se visualizan todos los controles ocultos.

Al igual que en las actividades tipo Gráficos, la captura de los datos acaba cuando el usuario decide grabarlos en un fichero de texto o pulsa el botón de Volver, en cuyo caso se comprueba si la conexión Bluetooth está activa para cancelarla en su caso.

Guardado de los datos:

Se codifica en todos los casos de igual manera en el método asociado a la pulsación del botón Guardar. Las operaciones que realiza son las siguientes:

- Detiene la representación de los datos
- Finaliza la conexión Bluetooth
- Abre o crea un fichero en la dirección correspondiente (en función de si se trata de ECG o EMG se crea en el directorio correspondiente), con el nombre proporcionado por el usuario en el EditText.
- A continuación, se van leyendo los datos del vector de datos y se van grabando como líneas de texto en el fichero. Como las primeras 1600 o 600 muestras eran nulas para crear el buffer (en función de la actividad donde se esté), estas no se graban en el fichero.
- Se cierra el fichero, se informa al usuario y se cierra la actividad, volviendo a la actividad anterior (Cardio o Miograma, según corresponda.

```
public void GuardarFichero() {
    onPause();
    if (servicio !=null){
        servicio.finalizarServicio();}

    nomarchivo= nomarchivo_text.getText().toString();
    fichero = new File(nomDirArch+nomarchivo);
        try {
        fis = new FileOutputStream(fichero);
        archivo = new OutputStreamWriter(fis);
        for(int k=1600;k<datos.size();k++) {
            String linea = datos.get(k).toString();
            archivo.write(linea + "\n");
        }
        archivo.flush();
        archivo.close();
    } catch (IOException e) {
    }
     try {
        archivo.close();
```

```
            } catch (IOException e) {
            }
            Toast t= Toast.makeText(this, "Los datos fueron grabados",
        Toast. LENGTH_SHORT);
            t.show();
            finish();
        }
```

Además, es necesario recordar añadir en el manifiesto el permiso para escribir en la memoria externa:

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

4.5.- Pruebas y verificación

Para el sistema final completo el siguiente esquema de pruebas:

- Creación de la conexión: Prueba de la creación del socket y envío de datos sencillo. Estos datos se visualizaron en alguno de los controles TextView utilizados.
- Prueba de representación de gráficos: Prueba del envío de la secuencia creciente de datos float (sistema de prueba sencillo Arduino) y representación de la rampa correspondiente en los gráficos.
- Prueba representación de ECG y EMG: Prueba de las distintas actividades con los datos enviados y capturados por el sistema Arduino.
- Prueba de guardado de ficheros: Creación de una aplicación simple que guarde en un fichero los datos de un objeto vector de floats.
- Prueba final de guardado: Prueba de guardado de los datos capturados en la aplicación definitiva.
- Pruebas finales: Prueba del sistema completo con cambios de actividades y opciones de manera aleatoria.

En estas pruebas se ha observado el correcto funcionamiento del sistema. No obstante, se ha de comentar que la pulsación de los controles de manera rápida, sobre todo la de finalización de actividad produce a veces el bloqueo de la actividad. Igualmente, como ya se ha comentado, si a la hora de establecer la conexión Bluetooth se tarda demasiado en pulsar el botón de *reset* de Arduino, la conexión no se puede establecer, no representándose nada. No obstante, se puede salir de la actividad y volver a entrar para reintentarlo.

En esta última fase se ha aprendido a realizar una conexión Bluetooth sencilla entre el sistema Arduino y una aplicación Android. Se ha conocido los parámetros básicos de configuración del módulo Bluetooth PRO basado en el chip BlueGiga. Se ha aprendido el uso de emuladores de puerto serie para Windows (Hterm) y Android (BT Simple Term). Se han iniciado en la utilización de manejadores para el intercambio de información entre hilos de una misma actividad. Finalmente, se ha continuado con el aprendizaje del manejo de archivos en Android con la grabación de ficheros de texto.

5

CONCLUSIONES Y LÍNEAS FUTURAS

Este último Capítulo contiene las conclusiones a las que se ha llegado tras la realización del TFG, así como una breve descripción de las posibles líneas de trabajo por las que se puede continuar desarrollando el trabajo realizado.

5.- Conclusiones y Líneas Futuras

En el presente TFG se ha presentado un sistema que permite capturar las señales de ECG y EMG mediante un sistema basado en Arduino y e-Health, para ser enviados vía Bluetooth a una aplicación móvil para Android donde dichas señales pueden ser representadas, almacenadas y utilizadas para dos experimentos simples, consistentes en la observación del ritmo cardiaco durante la audición de diversos estilos de música y la medición de la intensidad y el valor medio de los impulsos nerviosos del sistema muscular.

Como objetivo secundario y personal, se ha iniciado en el estudio de las aplicaciones basadas en Arduino y Android y en la comunicación entre ambas mediante tecnologías inalámbricas.

A lo largo del desarrollo del TFG se ha aprendido la configuración del *hardware* del Arduino ONE y de sus extensiones e-Health y Comunicaciones, con el módulo Bluetooth PRO. Se ha comprendido la estructura básica de sketch de programación Arduino y los fundamentos de su lenguaje de programación, trabajándose con las salidas digitales del mismo y su puerto serie. Se ha aprendido el manejo de librerías auxiliares y los principios básicos de configuración del chip de comunicaciones Bluetooth BlueGiga.

En lo que respecta a la programación para Android, se ha aprendido el manejo del Entorno de Desarrollo Android Studio y su configuración para el diseño de aplicaciones para diversos dispositivos. Se ha aprendido a utilizar un dispositivo físico real como dispositivo para la depuración del código mediante la facilidad de depuración USB. Se han diseñado diversos layouts para distintas actividades haciendo uso de diferentes tipos de controles, incluido uno externo al entorno de desarrollo, para la representación gráfica de las señales biológica. Debido a esto último, ha sido necesario comprender el funcionamiento de la dependencia XYPlot y la manera de incorporarla a un proyecto. Se ha aprendido el concepto de actividad, su ciclo de vida y el uso de varias actividades en una misma aplicación. Finalmente, se ha aprendido el concepto de thread o hilo dentro de una actividad y diversos modos de enviar información y actualizar controles entre varios hilos (método post y uso de manejadores o handles). Se han utilizado clases para la reproducción de ficheros de audio y para el manejo de ficheros.

En lo que respecta a la comunicación Bluetooth, se ha aprendido el uso de diversas aplicaciones para emular un puerto serie y enviar datos, como Hterm para PC o el BT Simple Terminal para Arduino. Se ha aprendido a configurar el chip BlueGiga para que realice una vinculación con otro dispositivo y se ha aprendido a realizar una conexión desde Android con un

dispositivo vinculado. Aunque no se ha utilizado finalmente, también se ha estudiado como realizar la vinculación vía *software* desde la propia aplicación.

Dado que el objetivo era adquirir un primer contacto con la tecnología Arduino y la programación Android, quedan diferentes aspectos de trabajo realizado que pueden ser mejorados o desarrollados con más profundidad. Estas líneas futuras se han dividido en tres tipos: Estéticos (pequeños cambios, fundamentalmente de apariencia), funcionales (extender o agregar las capacidades actuales manteniendo la estructura del sistema) y estructurales (que conllevan un cambio de la filosofía o modificaciones extremas del sistema).

Aspectos estéticos:

Por parte del sistema Arduino, se puede buscar un receptáculo adecuado y estético para el Arduino ONE y sus extensiones, de tal forma que quede encapsulado, solo con los conectores necesarios accesibles al usuario. Se puede mejorar la electrónica de conmutación entre la medida ECG y EMG, dejando en un único interruptor el control del puerto *hardware* de e-Health y el control del bucle software de realización de la medida y envío.

La aplicación Android puede ser mejorada en su interfaz, incluyendo controles personalizados diseñados exprofeso, permitir el funcionamiento en vertical y horizontal de la pantalla y en diferentes tamaños de dispositivos. Igualmente, nuevos tipos de representaciones gráficas más allá de AndroidPlot pueden ser buscadas para la representación del ECG y el EMG, con funciones de ampliación, corte, uso de marcadores, etc…

En lo que respecta a la comunicación Bluetooth, podría desconectarse y conectarse mediante un botón externo en el sistema Arduino o indicarse que la conexión ya se ha establecido mediante un led.

Aspectos funcionales:

En la parte Arduino, las principales funciones que pueden incluirse o extenderse serían, fundamentalmente y como ya se ha mencionado en el apartado estético, mecanizar o automatizar todo el sistema de selección de la medida ECG o EMG. Mediante el uso de interruptores controlables y usando las lógicas digitales del propio Arduino ONE se podría realizar mediante *software* la selección del bucle de medida en el sketch y realizar la función del puente selector. La selección de la medida podría incluso ser a través de algún comando enviado mediante Bluetooth por la aplicación Android. Los datos

podrían enviarse únicamente cuando el dispositivo esté configurado para medir (en caso contrario el sketch podría estar en un estado de espera). No obstante, esto entraría casi ya en cambios estructurales.

Por la parte Android, múltiples funciones nuevas pueden incorporarse. Por ejemplo, podría mejorarse el filtrado y procesado del ECG (algoritmo de Pan-Tompkins, umbrales ajustables dinámicamente, etc) y del EMG (mejora del filtro paso bajo, umbrales con histéresis ajustables, control de la ganancia vía bluetooth, etc) y el cálculo del ritmo cardiaco (mediante el promediado de varios intervalos entre latidos, p.e.). También puede mejorarse el sistema gestor de archivos para que el usuario pueda elegir donde almacenar los datos. A lo mencionado anteriormente respecto a la mejora de la representación gráfica, se podría incluir la posibilidad de almacenar los datos procesados (corte, marcadores, etc).

Respecto a la conexión Bluetooth, esta podría ser bidireccional, permitiendo el envío y recepción de datos por ambos componentes del sistema. De esta forma se podría configurar el estado del Arduino desde la aplicación Android. También se podría incluir en la aplicación Arduino el proceso de vinculación entre los dos dispositivos.

Aspectos arquitecturales:

En la parte Android, las extensiones de e-Health pueden diseñarse e implementarse de manera independiente, ya que la extensión provee de mucha más funcionalidad de la necesaria. Incluso se podría eliminar la placa de desarrollo Arduino ONE y hacer un diseño completo del sistema eliminando todos los componentes y módulos innecesarios. En el caso de disponer de una comunicación bidireccional, se puede replantear por completo la arquitectura del sketch utilizado, eliminando la necesidad de detener la adquisición de datos con un *reset* de la ejecución. En este caso, se podría crear un bucle de espera hasta que la conexión ya haya sido creada y, en ese caso, comenzar a enviar los datos. Otra solución al problema del establecimiento de la conexión sería usar un control de flujo de datos Hw, usando la señal del pin CTS de BlueGiga (pin que se pone a nivel bajo cuando el módulo está preparado para recibir datos) para controlar el bucle de adquisición de datos. Sin embargo, según parece entenderse del esquemático del módulo Bluetooth PRO, esta señal no se encuentra fácilmente accesible.

Igualmente, la parte Android también puede ser rediseñada desde cero. Una vez profundizado en los diversos aspectos de la programación Android, hay muchas partes de la programación utilizada que son redundantes o superfluos, con lo que un replanteamiento del código puede reducir el tamaño

de la aplicación. Así mismo, muchos de las configuraciones de los controles podrían gestionarse mediante el uso de recursos de estilo en XML, evitando la reiteración en el diseño de los layouts y la facilidad de adaptación a nuevos dispositivos.

Respecto al sistema de comunicaciones, este puede ser modificado a cualquier otro protocolo de comunicaciones, como Zigbee o WiFi.

6

REFERENCIAS

[1] “The Making of Arduino”, David Kushner (26 Oct 2011). IEEE Spectrum.

[2] Historia del Arduino. Arduinohistory.github.io.

[3] Web oficial Proyecto Arduino. https://www.arduino.cc/

[4] “Guia Básica de Arduino”, varios autores, Tiendaderobotica.com

[5] Documentación e-Health de Cooking Hacks, https://www.cooking-hacks.com/ documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical

[6] Documentación XBee Shield de Cooking Hacks, https://www.arduino.cc/en/Main/ArduinoXbeeShield

[7] Documentación Bluetooth PRO de Cooking Hacks, https://www.cooking-hacks.com/bluetooth-module-pro-for-arduino

[8] Web de apoyo al desarrollo en Android, https://developer.android.com

[9] “Usar y entender una actividad”, Ramón Invarato, http://jarroba.com/activity-entender-y-usar-una-actividad/

[10] “Bioseñales Médicas”, Daza Márquez, Alberto (Asignatura Optativa de Mención, Ingeniería Biomédica, 3º Curso, Apuntes y entregas personales)

[11] “Instrumentación Biomédica”, Navas González, Rafael (Asignatura Obligatoria de Mención, Ingeniería Biomédica, 3º Curso, Apuntes y entregas personales)

[12] Página oficial dependencia XYPlot, http://androidplot.com/

[13] Página descarga Hterm, http://www.der-hammer.info/terminal/

[14] Tutoriales Bluetooth PRO de Cooking Hacks, https://www.cooking-hacks.com/documentation/tutorials/bluetooth-module-pro-arduino-raspberry-pi-tutorial/

[15] Descarga BT Simple Term, https://play.google.com/store/apps/details?id=wingood.bluetooth.btsimpleterminal